Fine soccer(3)

Morio Kikuchi

Abstract：

We modify the present rule of soccer in order to be able to get points of plural kinds referring to the point system of rugby.

1. Probability(1)

We get a probability for a lead to become d points with ratio on score axis. In Figure 1, it is assumed that the score axis of one team is s_1 and the score axis of the other team is s_2. The probability for a lead of one team to become d points is the sum of products of the rates on the two points which are in the difference of d points in the figure.

$$P_l(d) = \sum_{j=0}^{\infty} P_{s_1}(j+d)P_{s_2}(j)$$

It is assumed that the mean of usual goals is 2.7 on one game namely 90 min.

```
#define tsum (90.)
#define mnt (90)
#define dmax (10)

#define lmd1a ((2.7/tsum)*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a ((2.7/tsum)*mnt)
#define lmd2b (0)
#define lmd2c (0)
#define lmd2d (0)
#define p2b (0)

#define ra 1
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
```

In this case

・rate of draw：0.176186
・mean score：2.699997
・modal score：2 (rate=0.244964)
・variance：2.699971

If we get a probability for a lead to become d points, using an alias $P_l(d) \equiv P(d)$, it is as follows:

```
d=-10  P(d)=0.000049
d= -9  P(d)=0.000192
d= -8  P(d)=0.000690
d= -7  P(d)=0.002237
d= -6  P(d)=0.006491
d= -5  P(d)=0.016662
d= -4  P(d)=0.037346
d= -3  P(d)=0.071989
d= -2  P(d)=0.117334
d= -1  P(d)=0.158903
d=  0  P(d)=0.176186
d=  1  P(d)=0.158903
d=  2  P(d)=0.117334
d=  3  P(d)=0.071989
d=  4  P(d)=0.037346
d=  5  P(d)=0.016662
d=  6  P(d)=0.006491
d=  7  P(d)=0.002237
d=  8  P(d)=0.000690
d=  9  P(d)=0.000192
d= 10  P(d)=0.000049
```

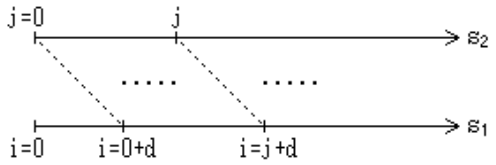

Figure 1

2. Probability(2)

We get a probability for one team to win in the case that the rest time is 30 min.

```
#define tsum (90.)
#define mnt (30)
#define dmax (10)

#define lmd1a ((2.7/tsum)*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a ((2.7/tsum)*mnt)
#define lmd2b (0)
#define lmd2c (0)
```

```
#define lmd2d (0)
#define p2b (0)

#define ra 1
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
```

With the above, using an alias $P_l(d) \equiv P(d)$, the probability for a lead to become d points in 30 min, for example is

```
d=-10  P(d)=0.000000
d= -9  P(d)=0.000000
d= -8  P(d)=0.000002
d= -7  P(d)=0.000017
d= -6  P(d)=0.000137
d= -5  P(d)=0.000930
d= -4  P(d)=0.005302
d= -3  P(d)=0.024495
d= -2  P(d)=0.086954
d= -1  P(d)=0.217726
d=  0  P(d)=0.328872
d=  1  P(d)=0.217726
d=  2  P(d)=0.086954
d=  3  P(d)=0.024495
d=  4  P(d)=0.005302
d=  5  P(d)=0.000930
d=  6  P(d)=0.000137
d=  7  P(d)=0.000017
d=  8  P(d)=0.000002
d=  9  P(d)=0.000000
d= 10  P(d)=0.000000
```

$$P_w(d) = P_l(-d) \times 0.5 + \sum_{j=1}^{-d+j \leq 10} P_l(-d+j)$$

With the above, using an alias $P_w(d) \equiv P(d)$, if one team leads by d points at that time, the probability for the team to win in the case that the rest time is 30 min is

```
d=-10  P(d)=0.000000
d= -9  P(d)=0.000000
d= -8  P(d)=0.000001
d= -7  P(d)=0.000010
d= -6  P(d)=0.000087
d= -5  P(d)=0.000621
d= -4  P(d)=0.003737
d= -3  P(d)=0.018636
```

```
d= -2  P(d)=0.074360
d= -1  P(d)=0.226700
d=  0  P(d)=0.500000
d=  1  P(d)=0.773299
d=  2  P(d)=0.925639
d=  3  P(d)=0.981363
d=  4  P(d)=0.996262
d=  5  P(d)=0.999378
d=  6  P(d)=0.999912
d=  7  P(d)=0.999989
d=  8  P(d)=0.999998
d=  9  P(d)=0.999999
d= 10  P(d)=0.999999
```

3. Probability(3)

In "1. Probability(1)", we assumed that lmd?a was constant. We show a connecting method in the case that lmd?a is not constant. We assume that lmd?a($\lambda_?$) is different between two 30 min in the rest time, 60 min like Figure 2. We assume two probabilities for one team to lead by d points to be $P_{l1}(d)$、$P_{l2}(d)$ respectively.

$d(\text{lmd1a1}) : i = -10, -9, ..., -2, -1, 0, 1, 2, ..., 9, 10$
$d(\text{lmd1a2}) : j = -10, -9, ..., -2, -1, 0, 1, 2, ..., 9, 10$

If we use a connection, a probability for one team to lead by 0 point, for example becomes as follows:

$$\begin{aligned} P_l(0)=&P_{l1}(-10)\times P_{l2}(10) + P_{l1}(-9)\times P_{l2}(9) + ... + P_{l1}(-2)\times P_{l2}(2) + P_{l1}(-1)\times P_{l2}(1) \\ &+P_{l1}(0)\times P_{l2}(0) \\ &+P_{l1}(1)\times P_{l2}(-1) + P_{l1}(2)\times P_{l2}(-2) + ... + P_{l1}(9)\times P_{l2}(-9) + P_{l1}(10)\times P_{l2}(-10) \end{aligned}$$

Therefore, generally

$$P_l(d) = \sum_{i=-10}^{i<=10,\ i+j=d} P_{l1}(i)\times P_{l2}(j) \qquad (j < -10 \Rightarrow j = -10,\ j > 10 \Rightarrow 10)$$

We use the following parameters, for example.

```
#define mnt 30

#define lmd1a1 (0.03*mnt)
#define lmd1a2 (0.015*mnt)

#define lmd2a1 (0.03*mnt)
#define lmd2a2 (0.03*1.3*mnt)
```

In this case, using an alias $P_l(d) \equiv P(d)$

```
d=-10  P(d)=0.000011
d= -9  P(d)=0.000066
```

```
d= -8  P(d)=0.000332
d= -7  P(d)=0.001431
d= -6  P(d)=0.005228
d= -5  P(d)=0.016243
d= -4  P(d)=0.042732
d= -3  P(d)=0.093203
d= -2  P(d)=0.162957
d= -1  P(d)=0.218234
d=  0  P(d)=0.211704
d=  1  P(d)=0.142327
d=  2  P(d)=0.069311
d=  3  P(d)=0.025855
d=  4  P(d)=0.007732
d=  5  P(d)=0.001920
d=  6  P(d)=0.000406
d=  7  P(d)=0.000075
d=  8  P(d)=0.000012
d=  9  P(d)=0.000002
d= 10  P(d)=0.000000
```

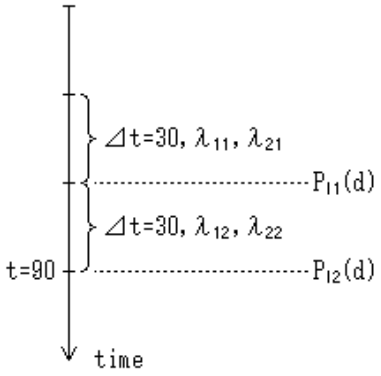

Figure 2

4. Probability(4)

We get a probability in which two teams draw being the same in the number of tries in rugby. First, we take the score axis on some number of tries in Figure 3 into consideration. The probability for two teams to become a tie in this number of tries is

$$P_{lh}(0) = \sum_{j=0}^{\infty} P_{s_1 h}(j) P_{s_2 h}(j)$$

Therefore, the probability to get is

$$P_l(0) = \sum_{h=0}^{\infty} P_{lh}(0)$$

We use the following parameters, for example.

```
#define tsum (80.)
#define mnt (80)
```

```
#define lmd1a ((6/tsum)*mnt)
#define lmd1b (0)
#define n1PK (3)
#define p1PK (0.75)
#define lmd1c (((n1PK*p1PK)/tsum)*mnt)
#define lmd1d (0)
#define p1b (0.7)

#define lmd2a ((6/tsum)*mnt)
#define lmd2b (0)
#define n2PK (3)
#define p2PK (0.75)
#define lmd2c (((n2PK*p2PK)/tsum)*mnt)
#define lmd2d (0)
#define p2b (0.7)

#define ra 5
#define rb 2
#define rc 3
#define rd 0
#define dgt 0
```

In this case

・rate of draw：0.017407
・mean score：45.149855
・modal score：41 (rate=0.026523)
・variance：271.039845

Therefore, hdraw/draw=0.007021/0.017407=0.403340 namely about 40 percent in draw games is the same in the number of tries.

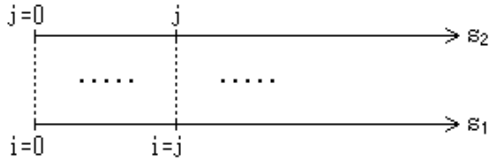

Figure 3

5. Time-out

In soccer, rugby, time-out is not built in. Basically, time-out is called when a ball and players do not have active qualification as we see in volleyball and so on. If we want to introduce time-out into soccer, rugby, out of such parts namely gaps, ones which satisfy the conditions that there is time to spare and so on need to be selected for call use. The following are the gaps which are suitable in my view.

・soccer：just after usual goal just before kick off
・rugby：just after conversion goal, penalty goal, drop goal just before drop kick
・soccer(Fine soccer)：just after goal caused by LK, MK, NK≡?K just before kick approach

・soccer：just after signal of goal kick, CK, DFK just before kick approach
・rugby：just after signal of penalty kick just before next play

On conversion goal, goal caused by ?K, failure case is included. The first two are gaps after scoring.

Figure 4, 5, 6 show timing to call time-out and kinds of horn in soccer. In the figures, we take time-out≡T.O.. Figure 7 shows timing to call time-out and kinds of horn in rugby. In the figure, we take conversion goal≡C.G., penalty goal≡P.G., drop goal≡D.G.. If time-out is called and it is signaled by referee, it begins. If the rest time of time-out becomes 10 sec, for example, 1st horn(short horn × 2) is sounded and if time-out ends, 2nd horn(medium horn × 1) is sounded. In Fine soccer, because we do selection between CK, DFK and ?K, the stream divides into the stream of Figure 5 and the stream of Figure 6 after selection like Figure 8 actually.

The time which is added to each half is $n_1 t_1$ assuming the size of time-out and the number of times to be t_1, n_1 respectively. We make time-out(t_1) 1 min, for example. Time-out can not be called continuously from among the gap.

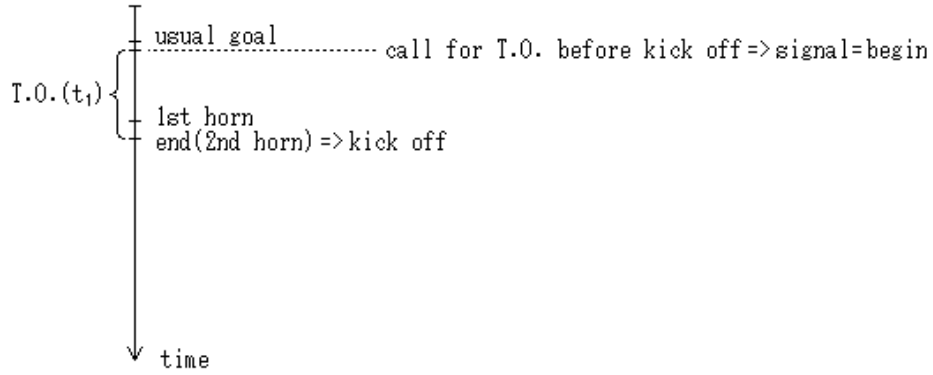

Figure 4

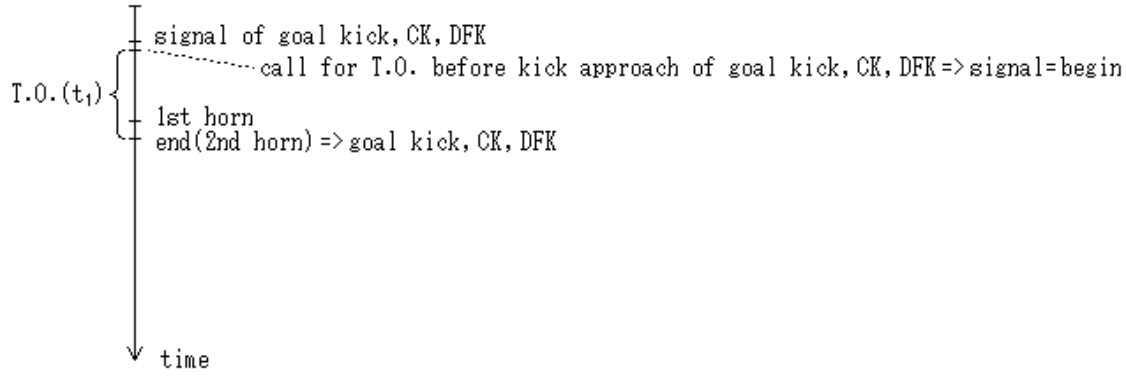

Figure 5

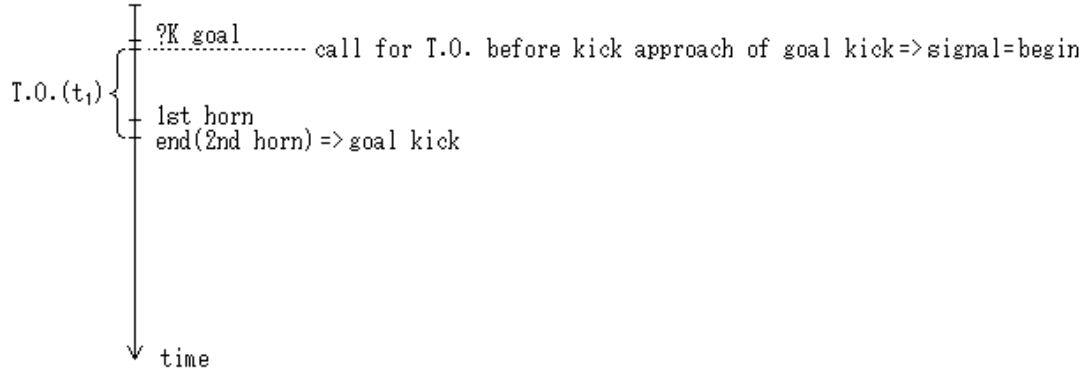

Figure 6

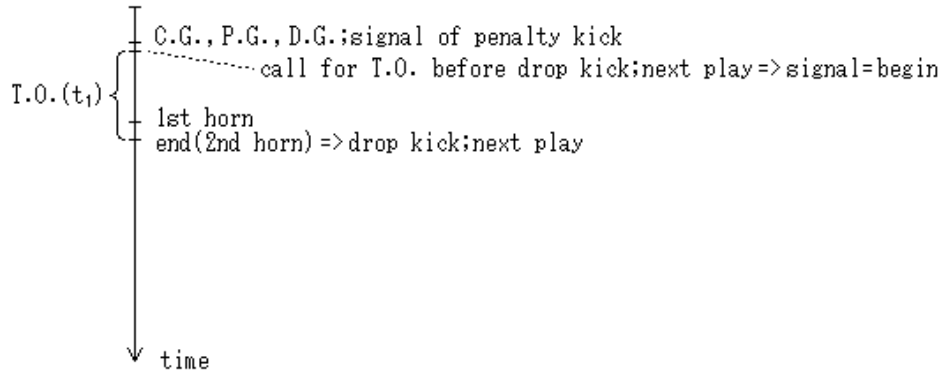

Figure 7

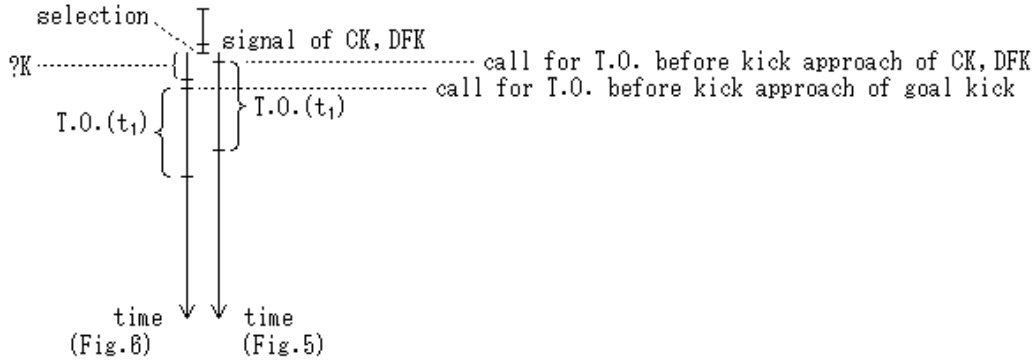

Figure 8

6. Break

In Figure 4, 5, 6, 7, time-out is called from among gap, however, the time size of gap is not constant. We introduce break in order to make the time from among which time-out can be called constant. Break begins if the following events occur and the signals are sended out.

- soccer : usual goal
- rugby : conversion goal, penalty goal, drop goal
- soccer(Fine soccer) : goal caused by LK, MK, NK≡?K

On conversion goal, goal caused by ?K, failure case is included. The first two are breaks after scoring.

Figure 9, 11 show the cases that time-out is not called from among break. The kind of horn is the same as Figure 4. If time-out is called from among break, it becomes like Figure 10, 12. In Fine soccer, because we do selection between CK, DFK and ?K, the stream divides into the stream of Figure 5 and the stream of Figure 12 after selection like Figure 13 actually.

The time which is added to each half is as follows:

$$\Delta t = n_1 t_1 + n_2 t_2 + n_3 t_3$$

t_1, n_1 are the size of time-out and the number of times respectively, t_2, n_2 are the size of break and the number of times in Figure 9 respectively and t_3, n_3 are the size of break and the number of times in Figure 11 respectively. We make time-out(t_1) 1 min, for example. We make break(t_2) in Figure 9 the size in which a simple rehydration can be done, 1 min, for example. On the other, we make break(t_3) in Figure 11 0.5 min, for example.

If we estimate the mean of the total number of breaks in Figure 9 per game roughly, it becomes as follows:

・soccer：2.5(usual goal) × 2=5
・rugby：(6(conversion goal) + 2(penalty goal)) × 2=16

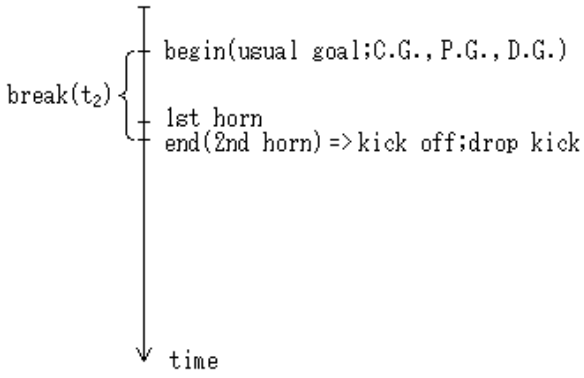

Figure 9

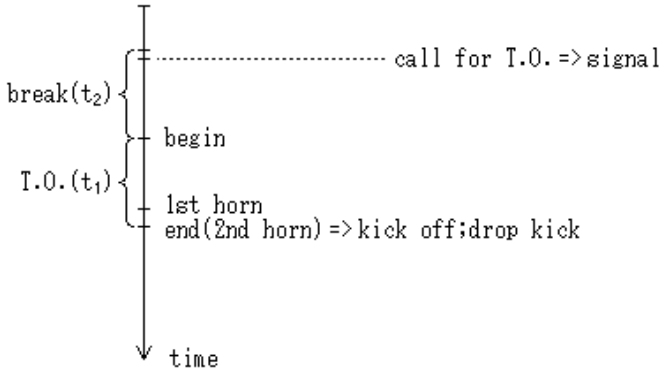

Figure 10

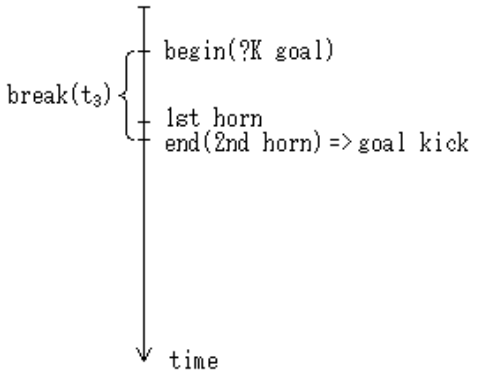

Figure 11

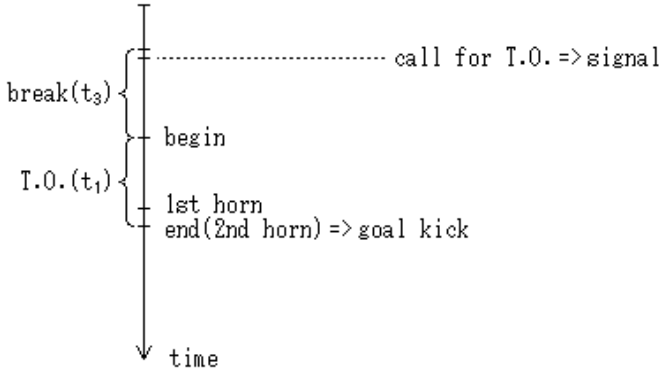

Figure 12

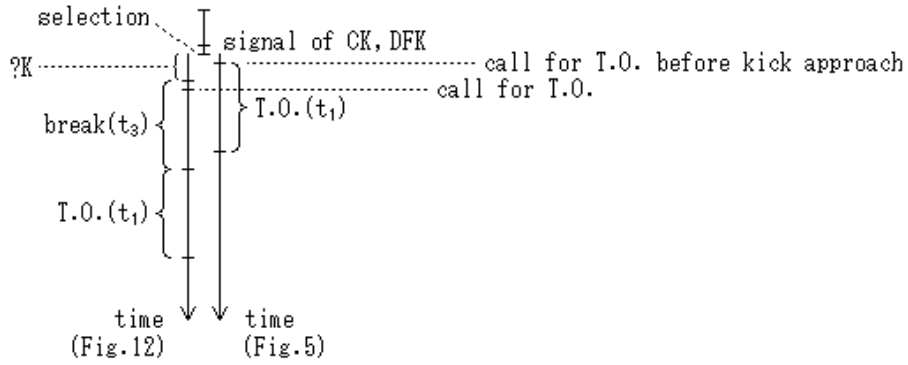

Figure 13

7. Routine

Referee, asistant referee are abbreviated to ref. and time keeper is abbreviated to t.k.. In Figure 4, 5, 6, 7, time-out is called from among gap. The detailed routine is as follows:

(1)manager(coach) calls for time-out from reserve referee
(2)just as reserve referee outside pitch carries a placard of time-out, horn(short horn × 3) is sounded
(3)ref. receives signal of (2)
(4)ref. sends out signal of time-out(sign with two arms : 'T' or 'O')
(5)t.k. receives signal of (4)
(6)t.k. starts clock of time-out
(7)t.k. sounds 1st horn(short horn × 2)
(8)t.k. sounds 2nd horn(medium horn × 1)

If ref. attaches twice short whistles in (4), reception of (5) becomes certain.

If an event which was shown in "6. Break" occurs and ref. sends out signal of the event, break is started automatically like the following(Figure 9, 11).

(1)ref. sends out signal of an event
(2)t.k. receives signal of (1)
(3)t.k. starts clock of break
(4)t.k. sounds 1st horn(short horn × 2)
(5)t.k. sounds 2nd horn(medium horn × 1)

In Figure 10, 12, time-out is called from among break. The detailed routine is as follows:

(1)manager(coach) calls for time-out from reserve referee
(2)just as reserve referee outside pitch carries a placard of time-out, horn(short horn × 3) is sounded
(3)ref. receives signal of (2)
(4)ref. sends out signal of time-out(sign with two arms : 'T' or 'O')
(5)t.k. receives signal of (4)
(6′)t.k. modifies clock of break
(7)t.k. sounds 1st horn(short horn × 2)
(8)t.k. sounds 2nd horn(medium horn × 1)

If ref. attaches twice short whistles in (4), reception of (5) becomes certain. If we replace (6) in the previously mentioned routine with (6′), the above routine is got. It is founded from the figure that the routine is extension of break by the size of time-out in substance. Naturally, horn in break which is the source of call is needless.

The time which is added to each half is as follows:

$$\Delta t = n_1 t_1 \qquad \text{(gap)}$$
$$\Delta t = n_1 t_1 + n_2 t_2 + n_3 t_3 \qquad \text{(break, break + gap)}$$

t.k. adds this Δt to each half. For example, we assume that it becomes $45 + 5$=50 min. If an additional time of 1 min takes place after 45 min, it becomes $45 + (5 + 1) = 51$ min.

If we use a software, the amount of time keeper's work can be lessen. There is no need to be conscious of sounding of horn, switching of indication between clock of game and clock of time-out, break, for example.

8. Transmission of call

We suppose the following two ways in transmission of call.

(A)manager(coach) ⇒ reserve referee ⇒ referee ⇒ time keeper
(B)manager(coach) ⇒ reserve referee ⇒ time keeper

(A) is the transmission of call which was mentioned in "7. Routine". In (B), time-out is taken not through referee. In this case, time keeper accesses clock if reserve referee sounds horn and the routine has no halfway (3), (4) as follows:

time-out from among gap
(1)manager(coach) calls for time-out from reserve referee
(2)just as reserve referee outside pitch carries a placard of time-out, horn(short horn × 3) is sounded
(5)t.k. receives signal of (2)
(6)t.k. starts clock of time-out
(7)t.k. sounds 1st horn(short horn × 2)
(8)t.k. sounds 2nd horn(medium horn × 1)

time-out from among break
(1)manager(coach) calls for time-out from reserve referee
(2)just as reserve referee outside pitch carries a placard of time-out, horn(short horn × 3) is sounded
(5)t.k. receives signal of (2)
(6′)t.k. modifies clock of break
(7)t.k. sounds 1st horn(short horn × 2)
(8)t.k. sounds 2nd horn(medium horn × 1)

If signal of (2) is not in the callable range in referee's view, referee sends out signal of reject(sign with two arms : 'X'). Besides, if signal of (2) is sended out during a kick approach, for example and kicker stops the kick because of this, a call can not be recognized from among the gap. (B) is suitable for time-out from among break.

There is an upper limit on the total number of time-outs which each team can call for. If the total number of time-outs of a team reaches a fixed number, reserve referee rejects the call. Manager(coach)

can not make a reservation for a call from reserve referee before gap, break.

9. The rest

If a remarkable delay action on kick of ball happens after time-out, break, we exchange offense and defense as a penalty. We change CK into goal kick, DFK into indirect free kick, goal kick after ?K into kick off and penalty kick into scrum.

As mentioned above, time-out from among gap is simpler than time-out from among break. Therefore, it is better that we introduce the latter after we become proficient in the basic structure doing the former a few years. In soccer

1st : gap(after scoring, goal kick, CK, DFK)
2nd : break(after scoring) + gap(goal kick, CK, DFK)

In rugby

1st : gap(after scoring, penalty kick)
2nd : break(after scoring) + gap(penalty kick)

****

ファインサッカー (3)

菊池盛雄

アブストラクト：
　ラグビーの得点体系を参考にして複数の得点が入るように現在のサッカーのルールを修正します。

1. 確率 (1)

　スコア軸上の割合から d 点リードとなる確率を求めましょう。図 1 において自チームのスコア軸を s_1、相手チームのスコア軸を s_2 とします。自チームが d 点リードとなる確率は図中の d 点差の二点における割合の積和です。

$$P_l(d) = \sum_{j=0}^{\infty} P_{s_1}(j+d)P_{s_2}(j)$$

一試合すなわち 90 分で通常のゴールの数の平均が 2.7 であるとします。

```
#define tsum (90.)
#define mnt (90)
#define dmax (10)

#define lmd1a ((2.7/tsum)*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a ((2.7/tsum)*mnt)
#define lmd2b (0)
#define lmd2c (0)
#define lmd2d (0)
#define p2b (0)

#define ra 1
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
```

この場合は

・引き分けの割合：0.176186
・平均スコア：2.699997
・最頻スコア：2 (割合=0.244964)
・分散：2.699971

となります。d 点リードとなる確率を求めると、$P_l(d) \equiv P(d)$ として以下のようになります。

```
d=-10  P(d)=0.000049
d= -9  P(d)=0.000192
d= -8  P(d)=0.000690
d= -7  P(d)=0.002237
d= -6  P(d)=0.006491
d= -5  P(d)=0.016662
d= -4  P(d)=0.037346
d= -3  P(d)=0.071989
d= -2  P(d)=0.117334
d= -1  P(d)=0.158903
d=  0  P(d)=0.176186
d=  1  P(d)=0.158903
d=  2  P(d)=0.117334
d=  3  P(d)=0.071989
d=  4  P(d)=0.037346
d=  5  P(d)=0.016662
d=  6  P(d)=0.006491
d=  7  P(d)=0.002237
d=  8  P(d)=0.000690
d=  9  P(d)=0.000192
d= 10  P(d)=0.000049
```

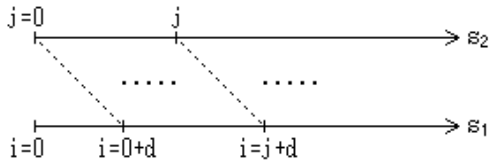

図 1

2. 確率 (2)

たとえば残り時間が 30 分である場合の自チームが勝つ確率を求めましょう。30 分で d 点リードとなる確率は

```
#define tsum (90.)
#define mnt (30)
#define dmax (10)

#define lmd1a ((2.7/tsum)*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a ((2.7/tsum)*mnt)
#define lmd2b (0)
```

```
#define lmd2c (0)
#define lmd2d (0)
#define p2b (0)

#define ra 1
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
```

より、$P_l(d) \equiv P(d)$ として

```
d=-10  P(d)=0.000000
d= -9  P(d)=0.000000
d= -8  P(d)=0.000002
d= -7  P(d)=0.000017
d= -6  P(d)=0.000137
d= -5  P(d)=0.000930
d= -4  P(d)=0.005302
d= -3  P(d)=0.024495
d= -2  P(d)=0.086954
d= -1  P(d)=0.217726
d=  0  P(d)=0.328872
d=  1  P(d)=0.217726
d=  2  P(d)=0.086954
d=  3  P(d)=0.024495
d=  4  P(d)=0.005302
d=  5  P(d)=0.000930
d=  6  P(d)=0.000137
d=  7  P(d)=0.000017
d=  8  P(d)=0.000002
d=  9  P(d)=0.000000
d= 10  P(d)=0.000000
```

残り 30 分における勝つ確率は、その時 d 点リードしていれば

$$P_w(d) = P_l(-d) \times 0.5 + \sum_{j=1}^{-d+j \leq 10} P_l(-d+j)$$

より、$P_w(d) \equiv P(d)$ として

```
d=-10  P(d)=0.000000
d= -9  P(d)=0.000000
d= -8  P(d)=0.000001
d= -7  P(d)=0.000010
d= -6  P(d)=0.000087
d= -5  P(d)=0.000621
d= -4  P(d)=0.003737
d= -3  P(d)=0.018636
```

```
d= -2  P(d)=0.074360
d= -1  P(d)=0.226700
d=  0  P(d)=0.500000
d=  1  P(d)=0.773299
d=  2  P(d)=0.925639
d=  3  P(d)=0.981363
d=  4  P(d)=0.996262
d=  5  P(d)=0.999378
d=  6  P(d)=0.999912
d=  7  P(d)=0.999989
d=  8  P(d)=0.999998
d=  9  P(d)=0.999999
d= 10  P(d)=0.999999
```

3. 確率 (3)

”1. 確率 (1)”では lmd?a は一定であるとしました。lmd?a が一定でない場合の連結方法を示します。図 2 のように残り時間が 60 分の内、二つの 30 分において lmd?a($\lambda_?$) が異なるとします。自チーム (lmd1a?) が d 点リードとなる確率を各々$P_{l1}(d)$、$P_{l2}(d)$ とします。

d(lmd1a1)：$i = -10, -9, ..., -2, -1, 0, 1, 2, ..., 9, 10$
d(lmd1a2)：$j = -10, -9, ..., -2, -1, 0, 1, 2, ..., 9, 10$

自チームがたとえば 0 点リードとなる確率は、連結により

$$\begin{aligned} P_l(0)=&P_{l1}(-10)\times P_{l2}(10) + P_{l1}(-9)\times P_{l2}(9) + ... + P_{l1}(-2)\times P_{l2}(2) + P_{l1}(-1)\times P_{l2}(1) \\ &+P_{l1}(0)\times P_{l2}(0) \\ &+P_{l1}(1)\times P_{l2}(-1) + P_{l1}(2)\times P_{l2}(-2) + ... + P_{l1}(9)\times P_{l2}(-9) + P_{l1}(10)\times P_{l2}(-10) \end{aligned}$$

したがって一般に

$$P_l(d) = \sum_{i=-10}^{i<=10,\ i+j=d} P_{l1}(i)\times P_{l2}(j) \qquad (j < -10 \Rightarrow j = -10,\ j > 10 \Rightarrow 10)$$

たとえば以下のようなパラメーターを用います。

```
#define mnt 30

#define lmd1a1 (0.03*mnt)
#define lmd1a2 (0.015*mnt)

#define lmd2a1 (0.03*mnt)
#define lmd2a2 (0.03*1.3*mnt)
```

この場合は、$P_l(d)$≡$P(d)$ として

```
d=-10  P(d)=0.000011
d= -9  P(d)=0.000066
d= -8  P(d)=0.000332
d= -7  P(d)=0.001431
```

```
d= -6  P(d)=0.005228
d= -5  P(d)=0.016243
d= -4  P(d)=0.042732
d= -3  P(d)=0.093203
d= -2  P(d)=0.162957
d= -1  P(d)=0.218234
d=  0  P(d)=0.211704
d=  1  P(d)=0.142327
d=  2  P(d)=0.069311
d=  3  P(d)=0.025855
d=  4  P(d)=0.007732
d=  5  P(d)=0.001920
d=  6  P(d)=0.000406
d=  7  P(d)=0.000075
d=  8  P(d)=0.000012
d=  9  P(d)=0.000002
d= 10  P(d)=0.000000
```

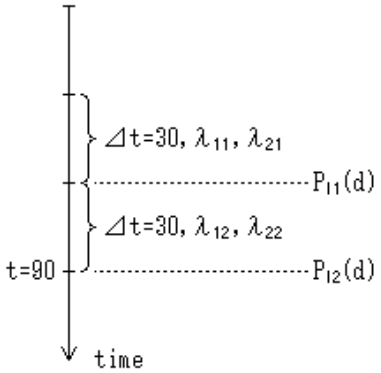

図 2

4. 確率 (4)

ラグビーにおいてトライ数が同じで引き分けになる確率を求めましょう。まず、図 3 のあるトライ数に関するスコア軸を考えます。このトライ数で同点になる確率は

$$P_{lh}(0) = \sum_{j=0}^{\infty} P_{s_1 h}(j) P_{s_2 h}(j)$$

したがって求める確率は

$$P_l(0) = \sum_{h=0}^{\infty} P_{lh}(0)$$

たとえば以下のようなパラメーターを用います。

```
#define tsum (80.)
#define mnt (80)

#define lmd1a ((6/tsum)*mnt)
#define lmd1b (0)
#define n1PK (3)
```

```
#define p1PK (0.75)
#define lmd1c (((n1PK*p1PK)/tsum)*mnt)
#define lmd1d (0)
#define p1b (0.7)

#define lmd2a ((6/tsum)*mnt)
#define lmd2b (0)
#define n2PK (3)
#define p2PK (0.75)
#define lmd2c (((n2PK*p2PK)/tsum)*mnt)
#define lmd2d (0)
#define p2b (0.7)

#define ra 5
#define rb 2
#define rc 3
#define rd 0
#define dgt 0
```

この場合は

・引き分けの割合：0.017407
・平均スコア：45.149855
・最頻スコア：41 (割合=0.026523)
・分散：271.039845

となります。したがって、引き分けの内、hdraw/draw=0.007021/0.017407=0.403340、すなわち 4 割ほどがトライ数が同じであるということになります。

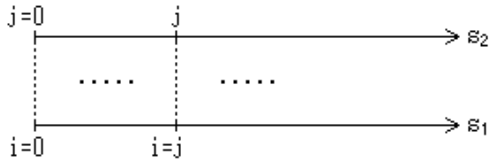

図 3

5. タイムアウト

サッカー、ラグビーではタイムアウトが組み込まれていません。そもそもタイムアウトというものはバレーボールなどでわかるように、ボール、プレーヤーにアクティブな資格がない時に call されるものです。サッカー、ラグビーにタイムアウトを組み込もうとすれば、そのような時間すなわちギャップの中から時間的に余裕がある等の条件を満たすものを call 用に選抜する必要があります。以下は筆者が適当であると思うギャップです。

・サッカー：通常のゴールの直後からキックオフの直前
・ラグビー：コンバージョンゴール、ペナルティゴール、ドロップゴールの直後からドロップキックの直前
・サッカー (ファインサッカー)：LK、MK、NK≡?K によるゴールの直後からキックアプローチの直前
・サッカー：ゴールキック、CK、DFK の signal の直後からキックアプローチの直前

・ラグビー：ペナルティキックの signal の直後から次のプレイの直前

コンバージョンゴール、?K によるゴールは失敗の場合も含みます。最初の二つは得点後のギャップです。

図 4、図 5、図 6 はサッカーにおけるタイムアウトを call するタイミングとホーンの種類を示しています。図中で、time-out≡T.O. としています。図 7 はラグビーにおけるタイムアウトを call するタイミングとホーンの種類を示しています。図中で、conversion goal≡C.G.、penalty goal≡P.G.、drop goal≡D.G. としています。タイムアウトが call され、レフェリーがこれを signal すればタイムアウトが開始されます。タイムアウトのたとえば 10 秒前になれば第一ホーン (ショートホーン× 2) を鳴らし、タイムアウトが終了すれば第二ホーン (メディアムホーン× 1) を鳴らします。ファインサッカーでは CK、DFK と?K のいずれかを選択するので、実際には図 8 のように selection の後は図 5 と図 6 のストリームに分岐します。

各ハーフに付加される時間は、タイムアウトの大きさと回数を t_1、n_1 として、$n_1 t_1$ です。タイムアウト (t_1) はたとえば 1 分とします。タイムアウトをギャップで連続して call することはできません。

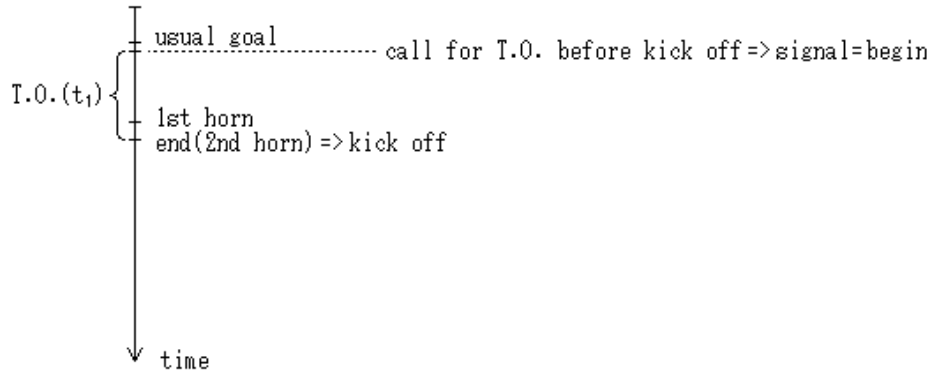

図 4

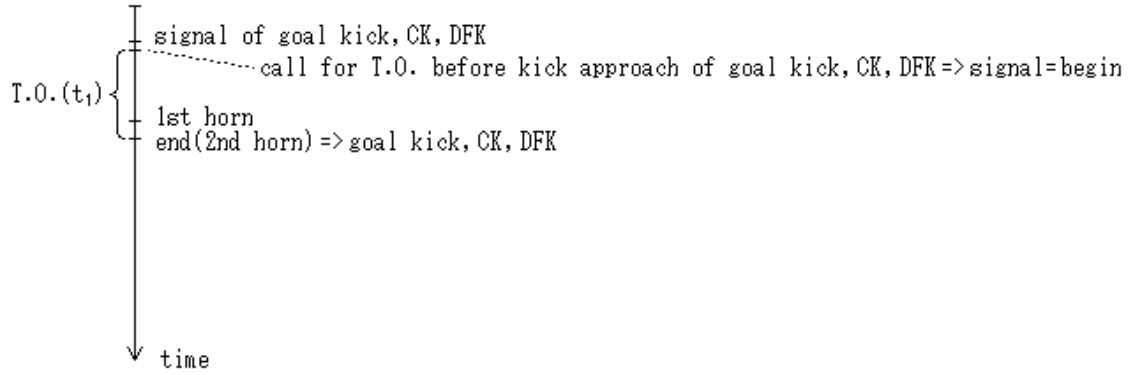

図 5

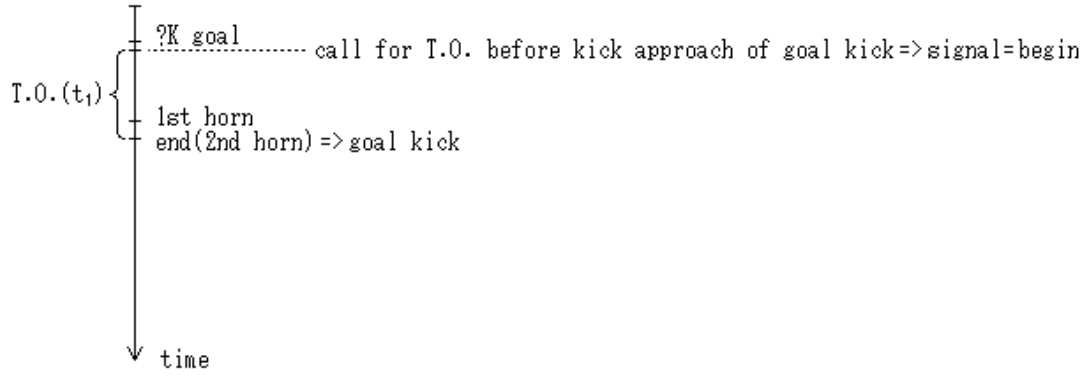

図 6

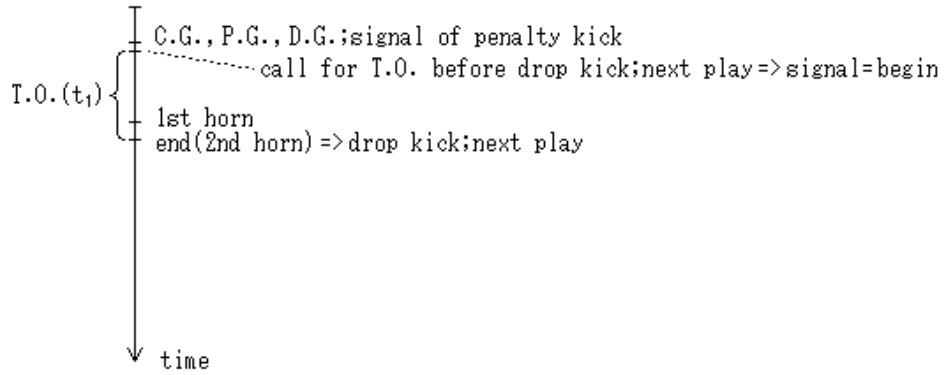

図 7

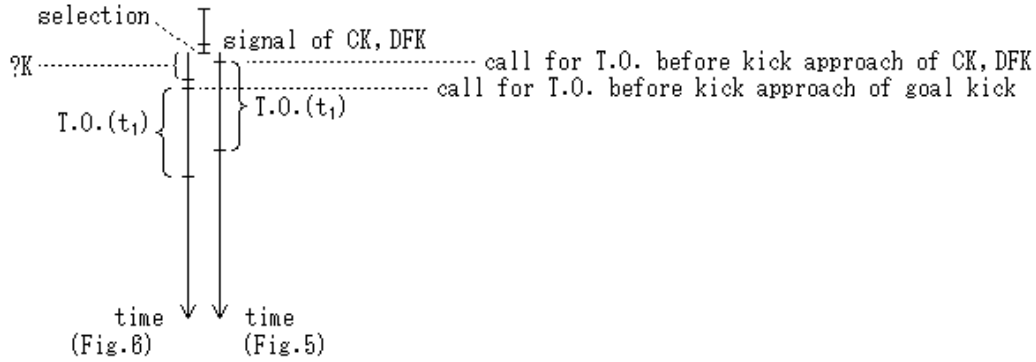

図 8

6. ブレイク

図 4、図 5、図 6、図 7 ではギャップからタイムアウトを call しますが、ギャップは一定ではありません。タイムアウトを call できる時間を一定とするためにブレイクを導入します。ブレイクは以下のイベントが発生し、その signal が発信されれば自動的に開始されます。

・サッカー：通常のゴール
・ラグビー：コンバージョンゴール、ペナルティゴール、ドロップゴール
・サッカー (ファインサッカー)：LK、MK、NK≡?K によるゴール

コンバージョンゴール、?K によるゴールは失敗の場合も含みます。最初の二つは得点後のブレイクです。

図 9、図 11 はブレイクにおいてタイムアウトを call しない場合を示しています。ホーンの種類は図 4 と同じです。ブレイクにおいてタイムアウトを call すれば図 10、図 12 のようになります。ファインサッカーでは CK、DFK と?K のいずれかを選択するので、実際には図 13 のように selection の後は図 5 と図 12 のストリームに分岐します。

各ハーフに付加される時間は以下のようになります。

$$\Delta t = n_1 t_1 + n_2 t_2 + n_3 t_3$$

t_1、n_1 はタイムアウトの大きさと回数、t_2、n_2 は図 9 におけるブレイクの大きさと回数、t_3、n_3 は図 11 におけるブレイクの大きさと回数です。タイムアウト (t_1) はたとえば 1 分とします。図 9 におけるブレイク (t_2) は簡易な水分補給ができる大きさ、たとえば 1 分とします。一方、図 11 におけるブレイク (t_3) はたとえば 0.5 分とします。

一試合当たりの図 9 におけるブレイクの総数の平均は荒く見積もると以下のようになります。

・サッカー：2.5(通常のゴール) × 2=5
・ラグビー：(6(コンバージョンゴール) ＋ 2(ペナルティゴール)) × 2=16

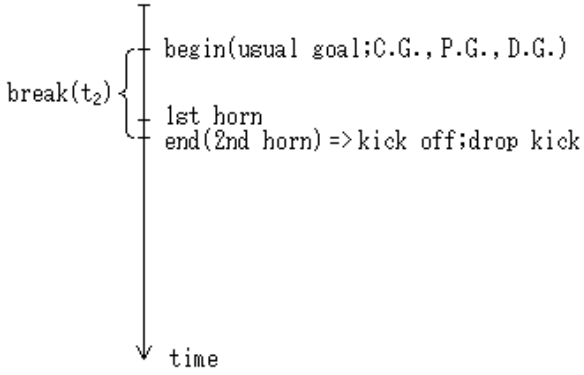

図 9

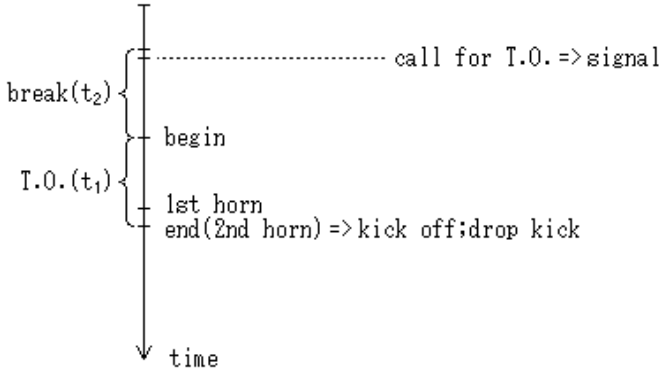

図 10

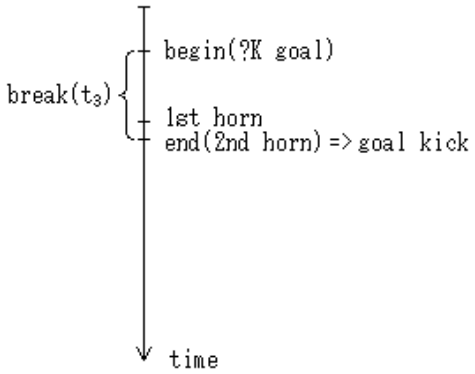

図 11

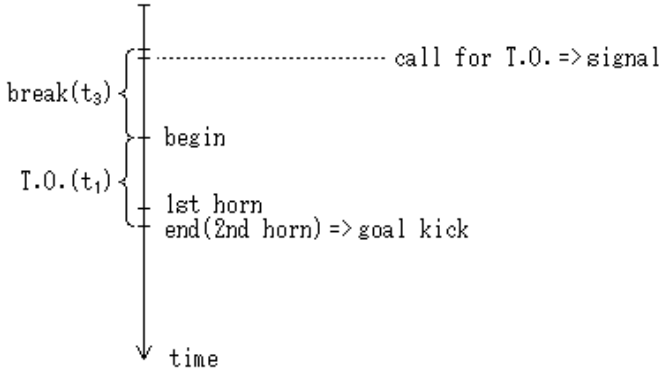

図 12

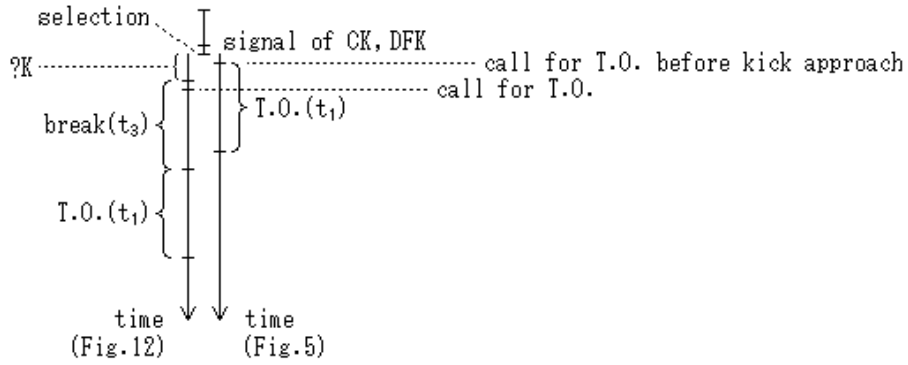

図 13

7. ルーチン

レフェリー、アシスタントレフェリーを ref.、タイムキーパーを t.k. と略称します。図 4、図 5、図 6、図 7 ではギャップからタイムアウトを call します。この詳細なルーチンは以下のようになります。

(1) 監督 (コーチ) がタイムアウトをリザーブレフェリーに call する
(2) ピッチ外にいるリザーブレフェリーがタイムアウトのプラカードを掲げると同時に、ホーン (ショートホーン× 3) を鳴らす
(3)ref. が (2) の signal を受信する
(4)ref. がタイムアウトの signal(腕文字：'T' または'O') を発信する
(5)t.k. が (4) の signal を受信する
(6)t.k. がタイムアウトの時計をスタートさせる
(7)t.k. が第一ホーン (ショートホーン× 2) を鳴らす
(8)t.k. が第二ホーン (メディアムホーン× 1) を鳴らす

(4) において ref. が二回の短いホイッスルを添えると (5) の受信が確実になります。

"6. ブレイク"で挙げたイベントが発生し、ref. がイベントの signal を発信すればブレイクが以下のように自動的に開始されます (図 9、図 11)。

(1)ref. がイベントの signal を発信する
(2)t.k. が (1) の signal を受信する
(3)t.k. がブレイクの時計をスタートさせる
(4)t.k. が第一ホーン (ショートホーン× 2) を鳴らす
(5)t.k. が第二ホーン (メディアムホーン× 1) を鳴らす

図 10、図 12 ではブレイクからタイムアウトを call します。この詳細なルーチンは以下のようになります。

(1) 監督 (コーチ) がタイムアウトをリザーブレフェリーに call する
(2) ピッチ外にいるリザーブレフェリーがタイムアウトのプラカードを掲げると同時に、ホーン (ショートホーン× 3) を鳴らす
(3)ref. が (2) の signal を受信する
(4)ref. がタイムアウトの signal(腕文字：'T' または'O') を発信する
(5)t.k. が (4) の signal を受信する
(6′)t.k. がブレイクの時計を修正する

(7)t.k. が第一ホーン (ショートホーン× 2) を鳴らす
(8)t.k. が第二ホーン (メディアムホーン× 1) を鳴らす

(4) において ref. が二回の短いホイッスルを添えると (5) の受信が確実になります。このルーチンは前述のルーチンにおいて (6) を (6′) に置き換えたものです。図からわかるように実質的にはブレイクの大きさがタイムアウトの大きさだけ延長されることになります。もちろん元のブレイクのホーンは不要です。
　各ハーフに付加される時間は

$\Delta t = n_1 t_1$　　(ギャップ)
$\Delta t = n_1 t_1 + n_2 t_2 + n_3 t_3$　　(ブレイク、ブレイク＋ギャップ)

です。t.k. はこの Δt を各ハーフに付加します。たとえば $45 + 5 = 50$ 分となったとします。もし 45 分過ぎに 1 分の付加時間が発生すれば、$45 + (5 + 1) = 51$ 分となります。
　ソフトウェアを利用すればタイムキーパーの仕事量を少なくすることができます。たとえば、ホーンを鳴らす、試合時計の表示とタイムアウトまたはブレイクの時計の表示を切り替える、は意識する必要がなくなります。

8. call の伝達
　call の伝達は以下の二通りを想定しています。

(A) 監督 (コーチ) ⇒ リザーブレフェリー ⇒ レフェリー ⇒ タイムキーパー
(B) 監督 (コーチ) ⇒ リザーブレフェリー ⇒ タイムキーパー

(A) は”7. ルーチン”で述べた call の伝達です。(B) ではレフェリーを介さずにタイムアウトをとります。この場合、タイムキーパーはリザーブレフェリーがホーンを鳴らせば時計にアクセスし、ルーチンは以下のように途中の (3)、(4) がありません。

ギャップからのタイムアウト：
(1) 監督 (コーチ) がタイムアウトをリザーブレフェリーに call する
(2) ピッチ外にいるリザーブレフェリーがタイムアウトのプラカードを掲げると同時に、ホーン (ショートホーン× 3) を鳴らす
(5)t.k. が (2) の signal を受信する
(6)t.k. がタイムアウトの時計をスタートさせる
(7)t.k. が第一ホーン (ショートホーン× 2) を鳴らす
(8)t.k. が第二ホーン (メディアムホーン× 1) を鳴らす

ブレイクからのタイムアウト：
(1) 監督 (コーチ) がタイムアウトをリザーブレフェリーに call する
(2) ピッチ外にいるリザーブレフェリーがタイムアウトのプラカードを掲げると同時に、ホーン (ショートホーン× 3) を鳴らす
(5)t.k. が (2) の signal を受信する
(6′)t.k. がブレイクの時計を修正する
(7)t.k. が第一ホーン (ショートホーン× 2) を鳴らす
(8)t.k. が第二ホーン (メディアムホーン× 1) を鳴らす

もし (2) の signal がレフェリーから見て call できる範囲に入っていなければ、レフェリーは reject の signal(腕文字：’X’) を発信します。そして、たとえばキックアプローチ中に (2) の signal があり、これが

原因でキッカーがキックを中止することがあれば、そのギャップでの call は認められません。(B) はブレイクからのタイムアウトに向いています。

各チームが call できるタイムアウトの総数には上限を設けます。各チームのタイムアウトの総数が規定数に達していればリザーブレフェリーは call を reject します。監督 (コーチ) がギャップとブレイクの前にリザーブレフェリーに call を予約することはできません。

9. その他

タイムアウト、ブレイク終了後にボールのキックに関する著しい遅延行為があれば、ペナルティとして攻守を替えます。CK はゴールキックに、DFK は間接フリーキックに、?K の後のゴールキックはキックオフに、ペナルティキックはスクラムに代えます。

前述のようにギャップからのタイムアウトの方がブレイクからのタイムアウトよりもシンプルです。したがって、まず前者を数年やって基本的な仕組みに習熟してから後者を導入する方がよいでしょう。サッカーでは

1st：ギャップ (得点後、ゴールキック、CK、DFK)
2nd：ブレイク (得点後) ＋ギャップ (ゴールキック、CK、DFK)

ラグビーでは

1st：ギャップ (得点後、ペナルティキック)
2nd：ブレイク (得点後) ＋ギャップ (ペナルティキック)

参考文献：

廣津　信義 ・吉井　秀邦 ・青葉　幸洋 ・吉村　雅文、
"時間内で得点を競う球技の試合における確率計算の方法"、
順天堂スポーツ健康科学研究、第１巻第３号、2010

****

List 1:connect.c
List 2:hdraw.c
List 3:horn.c
List 4:horn_cp.c

```
/* connect.c                                                              */
/* by Morio Kikuchi 2017.1.1                                              */
/* COMPILER:gpp(dos)                                                      */
/* COMMANDLINE:gcc -Dfar= -o connect.exe connect.c                        */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>

/*#define Rugby*/

#ifndef Rugby
/* Soccer */
#define tsum (90.)
#define mnt (90)
#define dmax (10)

#define lmd1a ((2.7/tsum)*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a ((2.7/tsum)*mnt)
#define lmd2b (0)
#define lmd2c (0)
#define lmd2d (0)
#define p2b (0)

#define ra 1
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
#else
/* Rugby ************************************************/
#define tsum (80.)
```

```
#define mnt (79.9)
#define dmax (60)

#define lmd1a ((6/tsum)*mnt)
#define lmd1b (0)
#define n1PK (3)
#define p1PK (0.75)
#define lmd1c (((n1PK*p1PK)/tsum)*mnt)
#define lmd1d (0)
#define p1b (0.7)

#define lmd2a ((6/tsum)*mnt)
#define lmd2b (0)
#define n2PK (3)
#define p2PK (0.75)
#define lmd2c (((n2PK*p2PK)/tsum)*mnt)
#define lmd2d (0)
#define p2b (0.7)

#define ra 5
#define rb 2
#define rc 3
#define rd 0
#define dgt 0
#endif

#define dmax_ (100)
#define break_9s (0.999999)
#define break9s  (0.9999)
#define READ 0
#define WRITE 1

int Umax,LKmax,MKmax,NKmax;
int lmt_Ph,lmt_Ps,lmt_Psum;
double lmd1a_,lmd1b_,lmd1c_,lmd1d_,lmd2a_,lmd2b_,lmd2c_,lmd2d_,mnt_;
double mgf,xg,yg,zg;
double P[(dmax+1)*2]/*[(int)(mnt/dt)+1]*/,P_[(dmax+1)*2],P_2[(dmax+1)*2];
double *Ph,*Qh,*Pi,*Qi,*Pj,*Qj,*Psum,*Qsum;
typedef struct {
int n;double P;} nP;
nP *Ps,*Qs;

FILE *fp;

double Pf(int,int,double,int);
double calc_w(int);
```

```
double calc_d(int);
void figs(int);


int main(int argc,unsigned char **argv)
{
int dargv;

mgf=pow(10,dgt);

fp=fopen("bbb.bin","wb");

mnt_=45;
lmd1a_=(0.015)*mnt_;
lmd1b_=(lmd1b/mnt)*mnt_;
lmd1c_=(lmd1c/mnt)*mnt_;
lmd1d_=(lmd1d/mnt)*mnt_;
lmd2a_=(0.012)*mnt_;
lmd2b_=(lmd2b/mnt)*mnt_;
lmd2c_=(lmd2c/mnt)*mnt_;
lmd2d_=(lmd2d/mnt)*mnt_;

for(dargv=-dmax;dargv<=dmax;dargv++){
xg=0;
figs(dargv);
Pf(0,dargv,xg,WRITE);
}
fprintf(fp," \n");



mnt_=45;
lmd1a_=(0.015)*mnt_;
lmd1b_=(lmd1b/mnt)*mnt_;
lmd1c_=(lmd1c/mnt)*mnt_;
lmd1d_=(lmd1d/mnt)*mnt_;
lmd2a_=(0.012)*mnt_;
lmd2b_=(lmd2b/mnt)*mnt_;
lmd2c_=(lmd2c/mnt)*mnt_;
lmd2d_=(lmd2d/mnt)*mnt_;

for(dargv=-dmax;dargv<=dmax;dargv++){
xg=0;
figs(dargv);
Pf(1,dargv,xg,WRITE);
}
```

```
fprintf(fp," \n");



for(dargv=-dmax;dargv<=dmax;dargv++){
xg=calc_d(dargv);
Pf(2,dargv,xg,WRITE);
fprintf(fp," d=%3d  p(d)=%f\n",dargv,xg);
}
fprintf(fp," \n");
return 0;

if(1){
/* 1+1 sections */
for(dargv=-dmax;dargv<=dmax;dargv++)
fprintf(fp," d=%3d  w(d)=%f\n",dargv,calc_w(dargv));
}
else{
/* (1+1)+1 sections */
for(dargv=-dmax;dargv<=dmax;dargv++) Pf(0,dargv,Pf(2,dargv,-1,READ),WRITE);

mnt_=30;
lmd1a_=(0.015)*mnt_;
lmd1b_=(lmd1b/mnt)*mnt_;
lmd1c_=(lmd1c/mnt)*mnt_;
lmd1d_=(lmd1d/mnt)*mnt_;
lmd2a_=(0.012)*mnt_;
lmd2b_=(lmd2b/mnt)*mnt_;
lmd2c_=(lmd2c/mnt)*mnt_;
lmd2d_=(lmd2d/mnt)*mnt_;

for(dargv=-dmax;dargv<=dmax;dargv++){
xg=0;
figs(dargv);
Pf(1,dargv,xg,WRITE);
}
fprintf(fp," \n");

for(dargv=-dmax;dargv<=dmax;dargv++){
xg=calc_d(dargv);
Pf(2,dargv,xg,WRITE);
fprintf(fp," d=%3d  p(d)=%f\n",dargv,xg);
}
fprintf(fp," \n");

fprintf(fp," te=%f\n",(double)(tsum-mnt));
```

```
for(dargv=-dmax;dargv<=dmax;dargv++)
fprintf(fp," d=%3d  w(d)=%f\n",dargv,calc_w(dargv));
}

fclose(fp);

return 0;
}/** main **/


double calc_w(int d)
{
int i,j;
static long count=0;
double val=0;

if(count==0){
for(i=-dmax;i<=dmax;i++){
val+=Pf(2,i,-1,READ);}
printf(" sum:%f\n",val/*Pf(2,i,-1,READ)*/);
}
count++;

val=0;
for(i=-d+1;i<=dmax;i++)
if(i>=-dmax && i<=dmax) val+=Pf(2,i,-1,READ);

val+=Pf(2,-d,-1,READ)*0.5;

return val;
}/** calc_w **/


double calc_d(int d)
{
int i,j;
double val=0;

for(i=-dmax;i<=dmax;i++)
/*for(j=-dmax;j<=dmax;j++){
if(i+j==d) val+=Pf(0,i,-1,READ)*Pf(1,j,-1,READ);
}*/
{
j=d-i;
if(j<-dmax) j=-dmax;
else if(j>dmax) j=dmax;
```

```
val+=Pf(0,i,-1,READ)*Pf(1,j,-1,READ);
}

return val;
}/** calc_d **/


double Pf(int flag,int i,double val,int RW)
{
/*int j;*/

/*j=t/dt;*/

if(flag==0){
if(RW==0) return P[dmax+i]/*[j]*/;
else              P[dmax+i]/*[j]*/=val;
}
else if(flag==1){
if(RW==0) return P_[dmax+i]/*[j]*/;
else              P_[dmax+i]/*[j]*/=val;
}
else{
if(RW==0) return P_2[dmax+i]/*[j]*/;
else              P_2[dmax+i]/*[j]*/=val;
}

return val;
}/** Pf **/


double times(int n)
{
int val;
double t;

if(n<=1) return 1.;
else{
t=n;val=n;
while(1){
val--;
t=t*val;
if(val<=1) break;
}

return t;
}
```

```
}/** times **/

int getmax(int flag,double lmd)
{
int k;
double val[2];

if(lmd>0){
val[0]=0;
for(k=0;;k++){
val[1]=exp(-lmd)*pow(lmd,k)/times(k);  /* Po:3 */
val[0]+=val[1];
if(flag==0) {if(val[0]>break9s) break;}
else        {if(val[0]>break_9s) break;}
}

/*printf(" lmd=%f k=%d\n",lmd,k);*/
return (k+2);
}

return 0;
}/** getmax **/


int mallocs(void)
{
int Umax_,LKmax_;

if(lmd1a>lmd2a)  Umax=getmax(1,lmd1a);else  Umax=getmax(1,lmd2a);
#ifndef Rugby
if(lmd1b>lmd2b) LKmax=getmax(1,lmd1b);else LKmax=getmax(1,lmd2b);
#endif
if(lmd1c>lmd2c) MKmax=getmax(1,lmd1c);else MKmax=getmax(1,lmd2c);
#ifndef Rugby
if(lmd1d>lmd2d) NKmax=getmax(1,lmd1d);else NKmax=getmax(1,lmd2d);
#endif

#ifndef Rugby
Umax_=LKmax;                        /* for LK */
LKmax_=LKmax;
#else
Umax_=Umax;
LKmax_=Umax;                        /* for conv. goal */
#endif

lmt_Ph=Umax_+1;
```

```
Ph=(double *)malloc(sizeof(double)*(lmt_Ph));
Qh=(double *)malloc(sizeof(double)*(lmt_Ph));
Pi=(double *)malloc(sizeof(double)*(MKmax+1));
Qi=(double *)malloc(sizeof(double)*(MKmax+1));
Pj=(double *)malloc(sizeof(double)*(NKmax+1));
Qj=(double *)malloc(sizeof(double)*(NKmax+1));

lmt_Ps=(int)((ra*Umax+rb*LKmax_+rc*MKmax+rd*NKmax+dmax_)*mgf);  /* +dmax_:for j=i+d */
Ps=(nP *)malloc(sizeof(nP)*(lmt_Ps));
if(Ps==NULL) {printf(" ?Ps:m\n");return 1;}
Qs=(nP *)malloc(sizeof(nP)*(lmt_Ps));
if(Qs==NULL) {printf(" ?Qs:m\n");return 1;}

lmt_Psum=Umax+LKmax_+MKmax+NKmax+1;
Psum=(double *)malloc(sizeof(double)*(lmt_Psum));
if(Psum==NULL) {printf(" ?Psum:m\n");return 1;}
Qsum=(double *)malloc(sizeof(double)*(lmt_Psum));
if(Qsum==NULL) {printf(" ?Qsum:m\n");return 1;}

return 0;
}/** mallocs **/


int reallocs(int flag)
{
int i,j,old;

if(flag==0){
old=lmt_Ph;lmt_Ph*=2;
Pj=(double *)realloc(Pj,sizeof(double)*(lmt_Ph));
}
else if(flag==1){
old=lmt_Ps;lmt_Ps*=2;
Ps=(nP *)realloc(Ps,sizeof(nP)*(lmt_Ps));
if(Ps==NULL) {printf(" ?Ps:r\n");return 1;}
Qs=(nP *)realloc(Qs,sizeof(nP)*(lmt_Ps));
if(Qs==NULL) {printf(" ?Qs:r\n");return 1;}

for(i=old;i<lmt_Ps;i++){
Ps[i].n=0;Ps[i].P=0;
Qs[i].n=0;Qs[i].P=0;
}
}
else{
old=lmt_Psum;lmt_Psum*=2;
```

```
Psum=(double *)realloc(Psum,sizeof(double)*(lmt_Psum));
if(Psum==NULL) {printf(" ?Psum:r\n");return 1;}
Qsum=(double *)realloc(Qsum,sizeof(double)*(lmt_Psum));
if(Qsum==NULL) {printf(" ?Qsum:r\n");return 1;}

for(i=old;i<lmt_Psum;i++){
Psum[i]=0;
Qsum[i]=0;
}
}

return 0;
}/** reallocs **/


void frees(void)
{
int j;

/*for(j=0;j<lmt_Ph;j++){
if(Pji[j]!=NULL) free(Pji[j]);
if(Qji[j]!=NULL) free(Qji[j]);
}
if(Pji!=NULL) free(Pji);
if(Qji!=NULL) free(Qji);*/

if(Ph!=NULL) free(Ph);
if(Qh!=NULL) free(Qh);
if(Pi!=NULL) free(Pi);
if(Qi!=NULL) free(Qi);
if(Pj!=NULL) free(Pj);
if(Qj!=NULL) free(Qj);

if(Ps!=NULL) free(Ps);
if(Qs!=NULL) free(Qs);

if(Psum!=NULL) free(Psum);
if(Qsum!=NULL) free(Qsum);
}/** frees **/


double Phij(int h,int i,int j,double vh,double vi,double vj,int RW)
{
if(RW==0){
return Ph[h]*Pi[i]*Pj[j];
}
```

```
else{
Ph[h]=vh;Pi[i]=vi;Pj[j]=vj;
}

return vh*vi*vj;
}/** Phij **/


double Qhij(int h,int i,int j,double vh,double vi,double vj,int RW)
{
if(RW==0){
return Qh[h]*Qi[i]*Qj[j];
}
else{
Qh[h]=vh;Qi[i]=vi;Qj[j]=vj;
}

return vh*vi*vj;
}/** Qhij **/


int get_Ps_S(void)
{
int h,i,j,k,endflag=0;
int score,ra_,rb_,rc_,rd_;
double val[8],wal[8];

ra_=ra*mgf;rb_=rb*mgf;rc_=rc*mgf;rd_=rd*mgf;

for(j=0;j<lmt_Psum;j++){
Psum[j]=0;
Qsum[j]=0;
}
for(j=0;j<lmt_Ps;j++){
Ps[j].n=0;Ps[j].P=0;
Qs[j].n=0;Qs[j].P=0;
}

val[4]=0;val[5]=0;val[6]=0;
wal[4]=0;wal[5]=0;wal[6]=0;

for(h=0;h<Umax+1;h++){
val[1]=exp(-lmd1a_)*pow(lmd1a_,h)/times(h);  /* Ps:7 */
wal[1]=exp(-lmd2a_)*pow(lmd2a_,h)/times(h);  /* Ps:7 */
val[7]+=val[1]*val[1];
wal[7]+=wal[1]*wal[1];
```

```
if(1){
/* LK is Poisson */

for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
Phij(i,j,k,0,0,0,WRITE);
Qhij(i,j,k,0,0,0,WRITE);
}

if((int)(rb*mgf)==0 && (int)(rc*mgf)==0 && (int)(rd*mgf)==0){
/*Pji[0][0]=val[1];Qji[0][0]=wal[1];*/
Phij(0,0,0,val[1],1,1,WRITE);
Qhij(0,0,0,wal[1],1,1,WRITE);
}
else if((int)(rb*mgf)==0 && (int)(rc*mgf)==0){
for(k=0;k<=NKmax;k++){
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(0,0,k,1,1,val[1]*val[4],WRITE);
Qhij(0,0,k,1,1,wal[1]*wal[4],WRITE);
}
}
else if((int)(rb*mgf)==0 && (int)(rd*mgf)==0){
for(j=0;j<=MKmax;j++){
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
Phij(0,j,0,1,val[1]*val[3],1,WRITE);
Qhij(0,j,0,1,wal[1]*wal[3],1,WRITE);
}
}
else if((int)(rc*mgf)==0 && (int)(rd*mgf)==0){
for(i=0;i<=LKmax;i++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
Phij(i,0,0,val[1]*val[2],1,1,WRITE);
Qhij(i,0,0,wal[1]*wal[2],1,1,WRITE);
}
}
else if((int)(rb*mgf)==0){
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
```

```
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(0,j,k,1,val[1]*val[3],val[4],WRITE);
Qhij(0,j,k,1,wal[1]*wal[3],wal[4],WRITE);
}
}
else if((int)(rc*mgf)==0){
for(i=0;i<=LKmax;i++)
for(k=0;k<=NKmax;k++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(i,0,k,val[1]*val[2],1,val[4],WRITE);
Qhij(i,0,k,wal[1]*wal[2],1,wal[4],WRITE);
}
}
else if((int)(rd*mgf)==0){
for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
Phij(i,j,0,val[1]*val[2],val[3],1,WRITE);
Qhij(i,j,0,wal[1]*wal[2],wal[3],1,WRITE);
}
}
else{
for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(i,j,k,val[1]*val[2],val[3],val[4],WRITE);
Qhij(i,j,k,wal[1]*wal[2],wal[3],wal[4],WRITE);
}
}

for(i=0;i<=LKmax;i++){
for(j=0;j<=MKmax;j++){
for(k=0;k<=NKmax;k++){
val[6]+=Phij(i,j,k,-1,-1,-1,READ)*Phij(i,j,k,-1,-1,-1,READ);
```

```
wal[6]+=Qhij(i,j,k,-1,-1,-1,READ)*Qhij(i,j,k,-1,-1,-1,READ);

while(1){
if(h+i+j>=lmt_Psum){
printf(" ?m_Psum:%d\n",lmt_Psum-(h+i+j));  /* ?m */
if(reallocs(2)) {endflag=1;goto next_mem;}
}
else break;
}

Psum[h+i+j+k]+=Phij(i,j,k,-1,-1,-1,READ);
Qsum[h+i+j+k]+=Qhij(i,j,k,-1,-1,-1,READ);

score=ra_*h+rb_*i+rc_*j+rd_*k;            /* important technique */
while(1){
if(score>=lmt_Ps){
printf(" ?m_Ps:%d\n",lmt_Ps-score);  /* ?m */
if(reallocs(1)) {endflag=1;goto next_mem;}
}
else break;
}

/*Ps[score].h=h;
Ps[score].i=i;
Ps[score].j=j;*/
if(Phij(i,j,k,-1,-1,-1,READ)>0) Ps[score].n++;
if(Qhij(i,j,k,-1,-1,-1,READ)>0) Qs[score].n++;
Ps[score].P+=Phij(i,j,k,-1,-1,-1,READ);
Qs[score].P+=Qhij(i,j,k,-1,-1,-1,READ);

val[5]+=Phij(i,j,k,-1,-1,-1,READ);
wal[5]+=Qhij(i,j,k,-1,-1,-1,READ);
if(val[5]>break_9s) {/*printf(" OK\n");*/goto next_S;}
if(wal[5]>break_9s) {/*printf(" OK\n");*/goto next_S;}
}/**for(k)**/
}/**for(j)**/
}/**for(i)**/
}/**if**/
else{
}/**else**/

/*projection(0+h*width,0,val[0]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,9);*/
}/**for(h)**/

next_S:
```

```
/*printf(" Soccer:h=%d",h);*/
/*printf(" draw_U=%f draw_min=%f",val[7],val[6]);*/
next_mem:

return endflag;
}/** get_Ps_S **/


int get_Ps_R(void)
{
int i,j,endflag=0;
int h,score,ra_,rb_,rc_;
double val[7],wal[7];

ra_=ra*mgf;rb_=rb*mgf;rc_=rc*mgf;

for(j=0;j<lmt_Psum;j++){
Psum[j]=0;
Qsum[j]=0;
}
for(j=0;j<lmt_Ps;j++){
Ps[j].n=0;Ps[j].P=0;
Qs[j].n=0;Qs[j].P=0;
}

val[4]=0;val[5]=0;val[6]=0;
wal[4]=0;wal[5]=0;wal[6]=0;

for(h=0;h<Umax+1;h++){
val[1]=exp(-lmd1a_)*pow(lmd1a_,h)/times(h);  /* Ps:7 */
wal[1]=exp(-lmd2a_)*pow(lmd2a_,h)/times(h);  /* Ps:7 */
val[6]+=val[1]*val[1];
wal[6]+=wal[1]*wal[1];

if(0){
}/**if**/
else{
if(h>=lmt_Ph){
printf(" ?m_Pj:%d\n",lmt_Ph-h);     /* ?m */
if(reallocs(0)) {endflag=1;goto next_mem;}
}

for(i=0;i<lmt_Ph;i++)
for(j=0;j<=MKmax;j++){
Phij(i,j,-0,0,0,0,WRITE);
Qhij(i,j,-0,0,0,0,WRITE);
```

```
}

if((int)(rb*mgf)==0 && (int)(rc*mgf)==0){
/*Pji[0][0]=val[1];Qji[0][0]=wal[1];*/
Phij(0,0,-0,val[1],1,1,WRITE);
Qhij(0,0,-0,wal[1],1,1,WRITE);
}
else if((int)(rb*mgf)==0){
for(j=0;j<=MKmax;j++){
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
/*Pji[0][j]=val[1]*val[3];
Qji[0][j]=wal[1]*wal[3];*/
Phij(0,j,-0,1,val[1]*val[3],1,WRITE);
Qhij(0,j,-0,1,wal[1]*wal[3],1,WRITE);
}
}
else if((int)(rc*mgf)==0){
for(i=0;i<=h;i++){
val[2]=(times(h)/(times(h-i)*times(i)))*pow(p1b,i)*pow(1-p1b,h-i);  /* B */
wal[2]=(times(h)/(times(h-i)*times(i)))*pow(p2b,i)*pow(1-p2b,h-i);  /* B */
/*Pji[i][0]=val[1]*val[2];
Qji[i][0]=wal[1]*wal[2];*/
Phij(i,0,-0,val[1]*val[2],1,1,WRITE);
Qhij(i,0,-0,wal[1]*wal[2],1,1,WRITE);
}
}
else{
for(i=0;i<=h;i++)
for(j=0;j<=MKmax;j++){
val[2]=(times(h)/(times(h-i)*times(i)))*pow(p1b,i)*pow(1-p1b,h-i);  /* B */
wal[2]=(times(h)/(times(h-i)*times(i)))*pow(p2b,i)*pow(1-p2b,h-i);  /* B */
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
Phij(i,j,-0,val[1]*val[2],val[3],1,WRITE);
Qhij(i,j,-0,wal[1]*wal[2],wal[3],1,WRITE);
}
}

for(i=0;i<=h;i++){
for(j=0;j<=MKmax;j++){
val[5]+=Phij(i,j,-0,-1,-1,-1,READ)*Phij(i,j,-0,-1,-1,-1,READ);
wal[5]+=Qhij(i,j,-0,-1,-1,-1,READ)*Qhij(i,j,-0,-1,-1,-1,READ);

while(1){
if(h+i+j>=lmt_Psum){
```

```
printf(" ?m_Psum:%d\n",lmt_Psum-(h+i+j));  /* ?m */
if(reallocs(2)) {endflag=1;goto next_mem;}
}
else break;
}

Psum[h+i+j]+=Phij(i,j,-0,-1,-1,-1,READ);
Qsum[h+i+j]+=Qhij(i,j,-0,-1,-1,-1,READ);

score=ra_*h+rb_*i+rc_*j;            /* important technique */
while(1){
if(score>=lmt_Ps){
printf(" ?m_Ps:%d\n",lmt_Ps-score);  /* ?m */
if(reallocs(1)) {endflag=1;goto next_mem;}
}
else break;
}

if(Phij(i,j,-0,-1,-1,-1,READ)>0) Ps[score].n++;
if(Qhij(i,j,-0,-1,-1,-1,READ)>0) Qs[score].n++;
Ps[score].P+=Phij(i,j,-0,-1,-1,-1,READ);
Qs[score].P+=Qhij(i,j,-0,-1,-1,-1,READ);

val[4]+=Phij(i,j,-0,-1,-1,-1,READ);
wal[4]+=Qhij(i,j,-0,-1,-1,-1,READ);
if(val[4]>break_9s) goto next_R;
if(wal[4]>break_9s) goto next_R;
}/**for(j)**/
}/**for(i)**/
}/**else**/

/*projection(0+h*width,0,val[0]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,9);*/
}/**for(h)**/

next_R:
/*printf(" Rugby:h=%d",h);*/
/*printf(" draw_U=%f draw_min=%f",val[6],val[5]);*/
next_mem:

return endflag;
}/** get_Ps_R **/


void figs(int d)
{
```

```
int i,j,dflag/*,d*/,fnum;
double lmd,val[7];

#ifndef Rugby
fnum=1;
#else
fnum=2;
#endif

if(mallocs()) goto end;
if(d>0)       dflag=1;
else if(d<0) {dflag=-1;d=-d;}
else          dflag=0;

if(abs(fnum)==1){
if(get_Ps_S()) goto end;

if(0){
lmd=lmd1a_+lmd1b_+lmd1c_+lmd1d_;
val[0]=0;
for(i=0;i<lmt_Psum;i++){
/*projection(0+i*width,0,Psum[i]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,10);*/
/*val[1]=exp(-lmd)*pow(lmd,i)/times(i);
printf("  t=%2d  %f  %f\n",i,sum[i],val[1]);*/
val[0]+=Psum[i];
if(val[0]>break9s) break;
}
/*printf(" t=%d S_Psum=%f\n",i,val[0]);*/
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;val[4]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
if(Ps[i].P>0) val[1]+=i*Ps[i].P;
if(Ps[i].P>val[3]) {val[2]=i;val[3]=Ps[i].P;}
val[4]+=Ps[i].P*Qs[i].P;
if(val[0]>break_9s) break;
}
/*printf(" draw=%f\n",val[4]);
printf(" mean=%f mode=%f(P=%f)\n",val[1]/mgf,val[2]/mgf,val[3]);*/

val[6]=val[1];

val[0]=0;/*val[1]=0;*/val[2]=0;
```

```
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
val[2]+=pow((val[6]-i)/mgf,2)*Ps[i].P;
if(val[0]>break_9s) break;
}
/*printf(" vrc=%f\n",val[2]);*/
}

if(0){
val[0]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
/*if(Po[i].n>0) printf("  %3d  %2d  %f\n",(int)(i/(double)mgf),Po[i].n,Po[i].P);*/
if(val[0]>break9s) break;
}
printf(" S_Ps[].P=%f at %d\n",val[0],i/(double)mgf);
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;
 /*dflag=-1;
d=1;*/
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;

/*99*/
if(dflag==0){

}/**if()**/
else if(dflag==1){

j=i-d;
if(i>=d) val[1]+=Ps[i].P*Qs[j].P;
}/**else if()**/
else{

j=i+d;
if(i>=0) val[1]+=Ps[i].P*Qs[j].P;
/*if(j>=lmt_Ps) printf(" %d\n",j-lmt_Ps);*/  /* -> +dmax_ */
}/**else**/

if(val[0]>break_9s) break;
}
if(dflag==0)      val[0]=val[4];
else if(dflag==1) val[0]=val[1];
else              val[0]=val[1];
```

```
fprintf(fp," figs:d=%3d  p(d)=%f\n",dflag*d,val[0]);
xg+=val[0];
}

/*printf(" mean_=%f\n",(double)(lmd1a_*ra+lmd1b_*rb+lmd1c_*rc+lmd1d_*rd));
printf(" ra=%f rb=%f rc=%f rd=%f\n",(double)ra,(double)rb,(double)rc,(double)rd);*/
}/**else if(fnum)**/
else if(abs(fnum)==2){
/*999*/
/* Rugby ************************************************/
if(get_Ps_R()) goto end;

if(1){
lmd=lmd1a_+lmd1b_+lmd1c_+lmd1d_;
val[0]=0;
for(i=0;i<lmt_Psum;i++){
/*projection(0+i*width,0,Psum[i]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,10);*/
/*val[1]=exp(-lmd)*pow(lmd,i)/times(i);
if((int)(rb*mgf)==0)
printf("  t=%2d  %f  %f\n",i,sum[i],val[1]);*/
val[0]+=Psum[i];
if(val[0]>break9s) break;
}
/*printf(" t=%d S_Psum=%f\n",i,val[0]);*/
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;val[4]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
if(Ps[i].P>0) val[1]+=i*Ps[i].P;
if(Ps[i].P>val[3]) {val[2]=i;val[3]=Ps[i].P;}
val[4]+=Ps[i].P*Qs[i].P;
if(val[0]>break_9s) break;
}
/*printf(" draw=%f\n",val[4]);
printf(" mean=%f mode=%f(P=%f)\n",val[1]/mgf,val[2]/mgf,val[3]);*/

val[6]=val[1];

val[0]=0;/*val[1]=0;*/val[2]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
val[2]+=pow((val[6]-i)/mgf,2)*Ps[i].P;
if(val[0]>break_9s) break;
```

```
}
/*printf(" vrc=%f\n",val[2]);*/
}

if(0){
val[0]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
/*if(Po[i].n>0) printf("  %3d  %2d  %f\n",(int)(i/(double)mgf),Po[i].n,Po[i].P);*/
if(val[0]>break9s) break;
}
printf(" S_Ps[].P=%f at %d\n",val[0],i/(double)mgf);
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;
 dflag=-1;
d=2;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;

/*99*/
if(dflag==0){

}/**if()**/
else if(dflag==1){

j=i-d;
if(i>=d) val[1]+=Ps[i].P*Qs[j].P;
}/**else if()**/
else{

j=i+d;
if(i>=0) val[1]+=Ps[i].P*Qs[j].P;
}/**else**/

if(val[0]>break_9s) break;
}
/*printf(" w=%f l=%f wld=%f\n",val[1],val[2],val[1]+val[2]+val[4]);*/
if(dflag==0)      val[0]=val[4];
else if(dflag==1) val[0]=val[1];
else              val[0]=val[1];
fprintf(fp," figs:d=%3d  p(d)=%f\n",dflag*d,val[0]);
xg+=val[0];
}
```

```
/*printf(" mean_=%f\n",(double)(lmd1a_*ra+(lmd1a_*p1b)*rb+lmd1c_*rc+lmd1d_*rd));
printf(" ra=%f rb=%f rc=%f rd=%f\n",(double)ra,(double)rb,(double)rc,(double)rd);*/
}/**else if(fnum)**/
else if(abs(fnum)==3){


}/**else if(fnum)**/

end:
frees();
}/** figs **/



/* hdraw.c                                                              */
/* by Morio Kikuchi 2017.1.1                                            */
/* COMPILER:gpp(dos)                                                    */
/* COMMANDLINE:gcc -Dfar= -o hdraw.exe hdraw.c                          */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>

#define Rugby

#ifndef Rugby
/* Soccer */
#define tsum (90.)
#define mnt (90)
#define dmax (10)

#define lmd1a ((2.7/tsum)*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a ((2.7/tsum)*mnt)
#define lmd2b (0)
#define lmd2c (0)
#define lmd2d (0)
#define p2b (0)

#define ra 1
```

```
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
#else
/* Rugby ************************************************/
#define tsum (80.)
#define mnt (80)
#define dmax (60)

#define lmd1a ((6/tsum)*mnt)
#define lmd1b (0)
#define n1PK (3)
#define p1PK (0.75)
#define lmd1c (((n1PK*p1PK)/tsum)*mnt)
#define lmd1d (0)
#define p1b (0.7)

#define lmd2a ((6/tsum)*mnt)
#define lmd2b (0)
#define n2PK (3)
#define p2PK (0.75)
#define lmd2c (((n2PK*p2PK)/tsum)*mnt)
#define lmd2d (0)
#define p2b (0.7)

#define ra 5
#define rb 2
#define rc 3
#define rd 0
#define dgt 0
#endif

#define dmax_ (100)
#define break_9s (0.999999)
#define break9s  (0.9999)
#define READ 0
#define WRITE 1

int Umax,LKmax,MKmax,NKmax;
int lmt_Ph,lmt_Ps,lmt_Psum;
double lmd1a_,lmd1b_,lmd1c_,lmd1d_,lmd2a_,lmd2b_,lmd2c_,lmd2d_,mnt_;
double mgf,xg,yg,zg;
double P[(dmax+1)*2]/*[(int)(mnt/dt)+1]*/,P_[(dmax+1)*2],P_2[(dmax+1)*2];
double *Ph,*Qh,*Pi,*Qi,*Pj,*Qj,*Psum,*Qsum;
typedef struct {
```

```
int n;double P;} nP;
nP *Ps,*Qs;

FILE *fp;

double Pf(int,int,double,int);
double calc_w(int);
double calc_d(int);
void figs(int);


int main(int argc,unsigned char **argv)
{
int dargv;

mgf=pow(10,dgt);

fp=fopen("bbb.bin","wb");

lmd1a_=lmd1a;
lmd1b_=lmd1b;
lmd1c_=lmd1c;
lmd1d_=lmd1d;
lmd2a_=lmd2a;
lmd2b_=lmd2b;
lmd2c_=lmd2c;
lmd2d_=lmd2d;

figs(dargv);

fclose(fp);

return 0;
}/** main **/


double calc_w(int d)
{
int i,j;
static long count=0;
double val=0;

if(count==0){
for(i=-dmax;i<=dmax;i++){
val+=Pf(2,i,-1,READ);}
printf(" sum:%f\n",val/*Pf(2,i,-1,READ)*/);
```

```
}
count++;

val=0;
for(i=-d+1;i<=dmax;i++)
if(i>=-dmax && i<=dmax) val+=Pf(2,i,-1,READ);

val+=Pf(2,-d,-1,READ)*0.5;

return val;
}/** calc_w **/


double calc_d(int d)
{
int i,j;
double val=0;

for(i=-dmax;i<=dmax;i++)
/*for(j=-dmax;j<=dmax;j++){
if(i+j==d) val+=Pf(0,i,-1,READ)*Pf(1,j,-1,READ);
}*/
{
j=d-i;
if(j<-dmax) j=-dmax;
else if(j>dmax) j=dmax;
val+=Pf(0,i,-1,READ)*Pf(1,j,-1,READ);
}

return val;
}/** calc_d **/


double Pf(int flag,int i,double val,int RW)
{
/*int j;*/

/*j=t/dt;*/

if(flag==0){
if(RW==0) return P[dmax+i]/*[j]*/;
else             P[dmax+i]/*[j]*/=val;
}
else if(flag==1){
if(RW==0) return P_[dmax+i]/*[j]*/;
else             P_[dmax+i]/*[j]*/=val;
```

```
}
else{
if(RW==0) return P_2[dmax+i]/*[j]*/;
else              P_2[dmax+i]/*[j]*/=val;
}

return val;
}/** Pf **/


double times(int n)
{
int val;
double t;

if(n<=1) return 1.;
else{
t=n;val=n;
while(1){
val--;
t=t*val;
if(val<=1) break;
}

return t;
}
}/** times **/

int getmax(int flag,double lmd)
{
int k;
double val[2];

if(lmd>0){
val[0]=0;
for(k=0;;k++){
val[1]=exp(-lmd)*pow(lmd,k)/times(k);  /* Po:3 */
val[0]+=val[1];
if(flag==0) {if(val[0]>break9s) break;}
else        {if(val[0]>break_9s) break;}
}

/*printf(" lmd=%f k=%d\n",lmd,k);*/
return (k+2);
}
```

```
return 0;
}/** getmax **/


int mallocs(void)
{
int Umax_,LKmax_;

if(lmd1a>lmd2a)  Umax=getmax(1,lmd1a);else  Umax=getmax(1,lmd2a);
#ifndef Rugby
if(lmd1b>lmd2b) LKmax=getmax(1,lmd1b);else LKmax=getmax(1,lmd2b);
#endif
if(lmd1c>lmd2c) MKmax=getmax(1,lmd1c);else MKmax=getmax(1,lmd2c);
#ifndef Rugby
if(lmd1d>lmd2d) NKmax=getmax(1,lmd1d);else NKmax=getmax(1,lmd2d);
#endif

#ifndef Rugby
Umax_=LKmax;                          /* for LK */
LKmax_=LKmax;
#else
Umax_=Umax;
LKmax_=Umax;                          /* for conv. goal */
#endif

lmt_Ph=Umax_+1;

Ph=(double *)malloc(sizeof(double)*(lmt_Ph));
Qh=(double *)malloc(sizeof(double)*(lmt_Ph));
Pi=(double *)malloc(sizeof(double)*(MKmax+1));
Qi=(double *)malloc(sizeof(double)*(MKmax+1));
Pj=(double *)malloc(sizeof(double)*(NKmax+1));
Qj=(double *)malloc(sizeof(double)*(NKmax+1));

lmt_Ps=(int)((ra*Umax+rb*LKmax_+rc*MKmax+rd*NKmax+dmax_)*mgf);  /* +dmax_:for j=i+d */
Ps=(nP *)malloc(sizeof(nP)*(lmt_Ps));
if(Ps==NULL) {printf(" ?Ps:m\n");return 1;}
Qs=(nP *)malloc(sizeof(nP)*(lmt_Ps));
if(Qs==NULL) {printf(" ?Qs:m\n");return 1;}

lmt_Psum=Umax+LKmax_+MKmax+NKmax+1;
Psum=(double *)malloc(sizeof(double)*(lmt_Psum));
if(Psum==NULL) {printf(" ?Psum:m\n");return 1;}
Qsum=(double *)malloc(sizeof(double)*(lmt_Psum));
if(Qsum==NULL) {printf(" ?Qsum:m\n");return 1;}
```

```
return 0;
}/** mallocs **/


int reallocs(int flag)
{
int i,j,old;

if(flag==0){
old=lmt_Ph;lmt_Ph*=2;
Pj=(double *)realloc(Pj,sizeof(double)*(lmt_Ph));
}
else if(flag==1){
old=lmt_Ps;lmt_Ps*=2;
Ps=(nP *)realloc(Ps,sizeof(nP)*(lmt_Ps));
if(Ps==NULL) {printf(" ?Ps:r\n");return 1;}
Qs=(nP *)realloc(Qs,sizeof(nP)*(lmt_Ps));
if(Qs==NULL) {printf(" ?Qs:r\n");return 1;}

for(i=old;i<lmt_Ps;i++){
Ps[i].n=0;Ps[i].P=0;
Qs[i].n=0;Qs[i].P=0;
}
}
else{
old=lmt_Psum;lmt_Psum*=2;
Psum=(double *)realloc(Psum,sizeof(double)*(lmt_Psum));
if(Psum==NULL) {printf(" ?Psum:r\n");return 1;}
Qsum=(double *)realloc(Qsum,sizeof(double)*(lmt_Psum));
if(Qsum==NULL) {printf(" ?Qsum:r\n");return 1;}

for(i=old;i<lmt_Psum;i++){
Psum[i]=0;
Qsum[i]=0;
}
}

return 0;
}/** reallocs **/


void frees(void)
{
int j;

/*for(j=0;j<lmt_Ph;j++){
```

```
if(Pji[j]!=NULL) free(Pji[j]);
if(Qji[j]!=NULL) free(Qji[j]);
}
if(Pji!=NULL) free(Pji);
if(Qji!=NULL) free(Qji);*/

if(Ph!=NULL) free(Ph);
if(Qh!=NULL) free(Qh);
if(Pi!=NULL) free(Pi);
if(Qi!=NULL) free(Qi);
if(Pj!=NULL) free(Pj);
if(Qj!=NULL) free(Qj);

if(Ps!=NULL) free(Ps);
if(Qs!=NULL) free(Qs);

if(Psum!=NULL) free(Psum);
if(Qsum!=NULL) free(Qsum);
}/** frees **/


double Phij(int h,int i,int j,double vh,double vi,double vj,int RW)
{
if(RW==0){
return Ph[h]*Pi[i]*Pj[j];
}
else{
Ph[h]=vh;Pi[i]=vi;Pj[j]=vj;
}

return vh*vi*vj;
}/** Phij **/


double Qhij(int h,int i,int j,double vh,double vi,double vj,int RW)
{
if(RW==0){
return Qh[h]*Qi[i]*Qj[j];
}
else{
Qh[h]=vh;Qi[i]=vi;Qj[j]=vj;
}

return vh*vi*vj;
}/** Qhij **/
```

```
int get_Ps_S(void)
{
int h,i,j,k,endflag=0;
int score,ra_,rb_,rc_,rd_;
double val[8],wal[8];

ra_=ra*mgf;rb_=rb*mgf;rc_=rc*mgf;rd_=rd*mgf;

for(j=0;j<lmt_Psum;j++){
Psum[j]=0;
Qsum[j]=0;
}
for(j=0;j<lmt_Ps;j++){
Ps[j].n=0;Ps[j].P=0;
Qs[j].n=0;Qs[j].P=0;
}

val[4]=0;val[5]=0;val[6]=0;
wal[4]=0;wal[5]=0;wal[6]=0;

for(h=0;h<Umax+1;h++){
val[1]=exp(-lmd1a_)*pow(lmd1a_,h)/times(h);  /* Ps:7 */
wal[1]=exp(-lmd2a_)*pow(lmd2a_,h)/times(h);  /* Ps:7 */
val[7]+=val[1]*val[1];
wal[7]+=wal[1]*wal[1];

if(1){
/* LK is Poisson */

for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
Phij(i,j,k,0,0,0,WRITE);
Qhij(i,j,k,0,0,0,WRITE);
}

if((int)(rb*mgf)==0 && (int)(rc*mgf)==0 && (int)(rd*mgf)==0){
/*Pji[0][0]=val[1];Qji[0][0]=wal[1];*/
Phij(0,0,0,val[1],1,1,WRITE);
Qhij(0,0,0,wal[1],1,1,WRITE);
}
else if((int)(rb*mgf)==0 && (int)(rc*mgf)==0){
for(k=0;k<=NKmax;k++){
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
```

```
Phij(0,0,k,1,1,val[1]*val[4],WRITE);
Qhij(0,0,k,1,1,wal[1]*wal[4],WRITE);
}
}
else if((int)(rb*mgf)==0 && (int)(rd*mgf)==0){
for(j=0;j<=MKmax;j++){
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
Phij(0,j,0,1,val[1]*val[3],1,WRITE);
Qhij(0,j,0,1,wal[1]*wal[3],1,WRITE);
}
}
else if((int)(rc*mgf)==0 && (int)(rd*mgf)==0){
for(i=0;i<=LKmax;i++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
Phij(i,0,0,val[1]*val[2],1,1,WRITE);
Qhij(i,0,0,wal[1]*wal[2],1,1,WRITE);
}
}
else if((int)(rb*mgf)==0){
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(0,j,k,1,val[1]*val[3],val[4],WRITE);
Qhij(0,j,k,1,wal[1]*wal[3],wal[4],WRITE);
}
}
else if((int)(rc*mgf)==0){
for(i=0;i<=LKmax;i++)
for(k=0;k<=NKmax;k++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(i,0,k,val[1]*val[2],1,val[4],WRITE);
Qhij(i,0,k,wal[1]*wal[2],1,wal[4],WRITE);
}
}
else if((int)(rd*mgf)==0){
for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
```

```
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
Phij(i,j,0,val[1]*val[2],val[3],1,WRITE);
Qhij(i,j,0,wal[1]*wal[2],wal[3],1,WRITE);
}
}
else{
for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
val[2]=exp(-lmd1b_)*pow(lmd1b_,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b_)*pow(lmd2b_,i)/times(i);  /* Ps:rb */
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
val[4]=exp(-lmd1d_)*pow(lmd1d_,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d_)*pow(lmd2d_,k)/times(k);  /* Ps:rd */
Phij(i,j,k,val[1]*val[2],val[3],val[4],WRITE);
Qhij(i,j,k,wal[1]*wal[2],wal[3],wal[4],WRITE);
}
}

for(i=0;i<=LKmax;i++){
for(j=0;j<=MKmax;j++){
for(k=0;k<=NKmax;k++){
val[6]+=Phij(i,j,k,-1,-1,-1,READ)*Phij(i,j,k,-1,-1,-1,READ);
wal[6]+=Qhij(i,j,k,-1,-1,-1,READ)*Qhij(i,j,k,-1,-1,-1,READ);

while(1){
if(h+i+j>=lmt_Psum){
printf(" ?m_Psum:%d\n",lmt_Psum-(h+i+j));  /* ?m */
if(reallocs(2)) {endflag=1;goto next_mem;}
}
else break;
}

Psum[h+i+j+k]+=Phij(i,j,k,-1,-1,-1,READ);
Qsum[h+i+j+k]+=Qhij(i,j,k,-1,-1,-1,READ);

score=ra_*h+rb_*i+rc_*j+rd_*k;            /* important technique */
while(1){
if(score>=lmt_Ps){
printf(" ?m_Ps:%d\n",lmt_Ps-score);  /* ?m */
if(reallocs(1)) {endflag=1;goto next_mem;}
}
else break;
```

```
}

/*Ps[score].h=h;
Ps[score].i=i;
Ps[score].j=j;*/
if(Phij(i,j,k,-1,-1,-1,READ)>0) Ps[score].n++;
if(Qhij(i,j,k,-1,-1,-1,READ)>0) Qs[score].n++;
Ps[score].P+=Phij(i,j,k,-1,-1,-1,READ);
Qs[score].P+=Qhij(i,j,k,-1,-1,-1,READ);

val[5]+=Phij(i,j,k,-1,-1,-1,READ);
wal[5]+=Qhij(i,j,k,-1,-1,-1,READ);
if(val[5]>break_9s) {/*printf(" OK\n");*/goto next_S;}
if(wal[5]>break_9s) {/*printf(" OK\n");*/goto next_S;}
}/**for(k)**/
}/**for(j)**/
}/**for(i)**/
}/**if**/
else{
}/**else**/

/*projection(0+h*width,0,val[0]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,9);*/
}/**for(h)**/

next_S:
/*printf(" Soccer:h=%d",h);*/
/*printf(" draw_U=%f draw_min=%f",val[7],val[6]);*/
next_mem:

return endflag;
}/** get_Ps_S **/



int get_Ps_R(void)
{
int i,j,i_,j_,endflag=0;
int h,score,score_,ra_,rb_,rc_;
double val[8],wal[7];

ra_=ra*mgf;rb_=rb*mgf;rc_=rc*mgf;

for(j=0;j<lmt_Psum;j++){
Psum[j]=0;
Qsum[j]=0;
}
```

```
for(j=0;j<lmt_Ps;j++){
Ps[j].n=0;Ps[j].P=0;
Qs[j].n=0;Qs[j].P=0;
}

val[4]=0;val[5]=0;val[6]=0;
wal[4]=0;wal[5]=0;wal[6]=0;
val[7]=0;

for(h=0;h<Umax+1;h++){
val[1]=exp(-lmd1a_)*pow(lmd1a_,h)/times(h);  /* Ps:7 */
wal[1]=exp(-lmd2a_)*pow(lmd2a_,h)/times(h);  /* Ps:7 */
val[6]+=val[1]*val[1];
wal[6]+=wal[1]*wal[1];

if(0){
}/**if**/
else{
if(h>=lmt_Ph){
printf(" ?m_Pj:%d\n",lmt_Ph-h);     /* ?m */
if(reallocs(0)) {endflag=1;goto next_mem;}
}

for(i=0;i<lmt_Ph;i++)
for(j=0;j<=MKmax;j++){
Phij(i,j,-0,0,0,0,WRITE);
Qhij(i,j,-0,0,0,0,WRITE);
}

if((int)(rb*mgf)==0 && (int)(rc*mgf)==0){
/*Pji[0][0]=val[1];Qji[0][0]=wal[1];*/
Phij(0,0,-0,val[1],1,1,WRITE);
Qhij(0,0,-0,wal[1],1,1,WRITE);
}
else if((int)(rb*mgf)==0){
for(j=0;j<=MKmax;j++){
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
/*Pji[0][j]=val[1]*val[3];
Qji[0][j]=wal[1]*wal[3];*/
Phij(0,j,-0,1,val[1]*val[3],1,WRITE);
Qhij(0,j,-0,1,wal[1]*wal[3],1,WRITE);
}
}
else if((int)(rc*mgf)==0){
for(i=0;i<=h;i++){
```

```
val[2]=(times(h)/(times(h-i)*times(i)))*pow(p1b,i)*pow(1-p1b,h-i);  /* B */
wal[2]=(times(h)/(times(h-i)*times(i)))*pow(p2b,i)*pow(1-p2b,h-i);  /* B */
/*Pji[i][0]=val[1]*val[2];
Qji[i][0]=wal[1]*wal[2];*/
Phij(i,0,-0,val[1]*val[2],1,1,WRITE);
Qhij(i,0,-0,wal[1]*wal[2],1,1,WRITE);
}
}
else{
for(i=0;i<=h;i++)
for(j=0;j<=MKmax;j++){
val[2]=(times(h)/(times(h-i)*times(i)))*pow(p1b,i)*pow(1-p1b,h-i);  /* B */
wal[2]=(times(h)/(times(h-i)*times(i)))*pow(p2b,i)*pow(1-p2b,h-i);  /* B */
val[3]=exp(-lmd1c_)*pow(lmd1c_,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c_)*pow(lmd2c_,j)/times(j);  /* Ps:rc */
Phij(i,j,-0,val[1]*val[2],val[3],1,WRITE);
Qhij(i,j,-0,wal[1]*wal[2],wal[3],1,WRITE);
}
}

for(i=0;i<=h;i++){
for(j=0;j<=MKmax;j++){
val[5]+=Phij(i,j,-0,-1,-1,-1,READ)*Phij(i,j,-0,-1,-1,-1,READ);
wal[5]+=Qhij(i,j,-0,-1,-1,-1,READ)*Qhij(i,j,-0,-1,-1,-1,READ);

while(1){
if(h+i+j>=lmt_Psum){
printf(" ?m_Psum:%d\n",lmt_Psum-(h+i+j));  /* ?m */
if(reallocs(2)) {endflag=1;goto next_mem;}
}
else break;
}

Psum[h+i+j]+=Phij(i,j,-0,-1,-1,-1,READ);
Qsum[h+i+j]+=Qhij(i,j,-0,-1,-1,-1,READ);

score=ra_*h+rb_*i+rc_*j;            /* important technique */
while(1){
if(score>=lmt_Ps){
printf(" ?m_Ps:%d\n",lmt_Ps-score);  /* ?m */
if(reallocs(1)) {endflag=1;goto next_mem;}
}
else break;
}

if(Phij(i,j,-0,-1,-1,-1,READ)>0) Ps[score].n++;
```

```
if(Qhij(i,j,-0,-1,-1,-1,READ)>0) Qs[score].n++;
Ps[score].P+=Phij(i,j,-0,-1,-1,-1,READ);
Qs[score].P+=Qhij(i,j,-0,-1,-1,-1,READ);

val[4]+=Phij(i,j,-0,-1,-1,-1,READ);
wal[4]+=Qhij(i,j,-0,-1,-1,-1,READ);
if(val[4]>break_9s) goto next_R;
if(wal[4]>break_9s) goto next_R;
}/**for(j)**/
}/**for(i)**/

/* hdraw */
for(i=0;i<=h;i++)for(j=0;j<=MKmax;j++){
score=ra_*h+rb_*i+rc_*j;
for(i_=0;i_<=h;i_++)for(j_=0;j_<=MKmax;j_++){
score_=ra_*h+rb_*i_+rc_*j_;
if(score==score_) val[7]+=Phij(i,j,-0,-1,-1,-1,READ)*Qhij(i_,j_,-0,-1,-1,-1,READ);
}
}
}/**else**/

/*projection(0+h*width,0,val[0]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,9);*/
}/**for(h)**/

next_R:
yg=val[7];
/*printf(" Rugby:h=%d",h);*/
/*printf(" draw_U=%f draw_min=%f",val[6],val[5]);*/
next_mem:

return endflag;
}/** get_Ps_R **/


void figs(int d)
{
int i,j,dflag/*,d*/,fnum;
double lmd,val[7];

#ifndef Rugby
fnum=1;
#else
fnum=2;
#endif
```

```
if(mallocs()) goto end;
if(d>0)       dflag=1;
else if(d<0) {dflag=-1;d=-d;}
else          dflag=0;

if(abs(fnum)==1){
if(get_Ps_S()) goto end;

if(0){
lmd=lmd1a_+lmd1b_+lmd1c_+lmd1d_;
val[0]=0;
for(i=0;i<lmt_Psum;i++){
/*projection(0+i*width,0,Psum[i]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,10);*/
/*val[1]=exp(-lmd)*pow(lmd,i)/times(i);
printf("  t=%2d  %f  %f\n",i,sum[i],val[1]);*/
val[0]+=Psum[i];
if(val[0]>break9s) break;
}
/*printf(" t=%d S_Psum=%f\n",i,val[0]);*/
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;val[4]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
if(Ps[i].P>0) val[1]+=i*Ps[i].P;
if(Ps[i].P>val[3]) {val[2]=i;val[3]=Ps[i].P;}
val[4]+=Ps[i].P*Qs[i].P;
if(val[0]>break_9s) break;
}
/*printf(" draw=%f\n",val[4]);
printf(" mean=%f mode=%f(P=%f)\n",val[1]/mgf,val[2]/mgf,val[3]);*/

val[6]=val[1];

val[0]=0;/*val[1]=0;*/val[2]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
val[2]+=pow((val[6]-i)/mgf,2)*Ps[i].P;
if(val[0]>break_9s) break;
}
/*printf(" vrc=%f\n",val[2]);*/
}

if(0){
```

```
val[0]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
/*if(Po[i].n>0) printf("  %3d  %2d  %f\n",(int)(i/(double)mgf),Po[i].n,Po[i].P);*/
if(val[0]>break9s) break;
}
printf(" S_Ps[].P=%f at %d\n",val[0],i/(double)mgf);
}

if(0){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;
 /*dflag=-1;
d=1;*/
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;

/*99*/
if(dflag==0){

}/**if()**/
else if(dflag==1){

j=i-d;
if(i>=d) val[1]+=Ps[i].P*Qs[j].P;
}/**else if()**/
else{

j=i+d;
if(i>=0) val[1]+=Ps[i].P*Qs[j].P;
/*if(j>=lmt_Ps) printf(" %d\n",j-lmt_Ps);*/  /* -> +dmax_ */
}/**else**/

if(val[0]>break_9s) break;
}
if(dflag==0)      val[0]=val[4];
else if(dflag==1) val[0]=val[1];
else              val[0]=val[1];
fprintf(fp," figs:d=%3d  p(d)=%f\n",dflag*d,val[0]);
xg+=val[0];
}

/*printf(" mean_=%f\n",(double)(lmd1a_*ra+lmd1b_*rb+lmd1c_*rc+lmd1d_*rd));
printf(" ra=%f rb=%f rc=%f rd=%f\n",(double)ra,(double)rb,(double)rc,(double)rd);*/
}/**else if(fnum)**/
else if(abs(fnum)==2){
/*999*/
```

```
/* Rugby ************************************************/
if(get_Ps_R()) goto end;

if(1){
lmd=lmd1a_+lmd1b_+lmd1c_+lmd1d_;
val[0]=0;
for(i=0;i<lmt_Psum;i++){
/*projection(0+i*width,0,Psum[i]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,10);*/
/*val[1]=exp(-lmd)*pow(lmd,i)/times(i);
if((int)(rb*mgf)==0)
printf("  t=%2d  %f  %f\n",i,sum[i],val[1]);*/
val[0]+=Psum[i];
if(val[0]>break9s) break;
}
/*printf(" t=%d S_Psum=%f\n",i,val[0]);*/
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;val[4]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
if(Ps[i].P>0) val[1]+=i*Ps[i].P;
if(Ps[i].P>val[3]) {val[2]=i;val[3]=Ps[i].P;}
val[4]+=Ps[i].P*Qs[i].P;
if(val[0]>break_9s) break;
}
printf(" draw=%f hdraw=%f hdraw/draw=%f\n",val[4],yg,yg/val[4]);
/*printf(" draw=%f\n",val[4]);*/
printf(" mean=%f mode=%f(P=%f)\n",val[1]/mgf,val[2]/mgf,val[3]);

val[6]=val[1];

val[0]=0;/*val[1]=0;*/val[2]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
val[2]+=pow((val[6]-i)/mgf,2)*Ps[i].P;
if(val[0]>break_9s) break;
}
printf(" vrc=%f\n",val[2]);
}

if(0){
val[0]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
```

```
/*if(Po[i].n>0) printf("  %3d  %2d  %f\n",(int)(i/(double)mgf),Po[i].n,Po[i].P);*/
if(val[0]>break9s) break;
}
printf(" S_Ps[].P=%f at %d\n",val[0],i/(double)mgf);
}

if(0){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;
 /*dflag=-1;
d=2;*/
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;

/*99*/
if(dflag==0){

}/**if()**/
else if(dflag==1){

j=i-d;
if(i>=d) val[1]+=Ps[i].P*Qs[j].P;
}/**else if()**/
else{

j=i+d;
if(i>=0) val[1]+=Ps[i].P*Qs[j].P;
}/**else**/

if(val[0]>break_9s) break;
}
/*printf(" w=%f l=%f wld=%f\n",val[1],val[2],val[1]+val[2]+val[4]);*/
if(dflag==0)      val[0]=val[4];
else if(dflag==1) val[0]=val[1];
else              val[0]=val[1];
fprintf(fp," figs:d=%3d  p(d)=%f\n",dflag*d,val[0]);
xg+=val[0];
}

/*printf(" mean_=%f\n",(double)(lmd1a_*ra+(lmd1a_*p1b)*rb+lmd1c_*rc+lmd1d_*rd));
printf(" ra=%f rb=%f rc=%f rd=%f\n",(double)ra,(double)rb,(double)rc,(double)rd);*/
}/**else if(fnum)**/
else if(abs(fnum)==3){


}/**else if(fnum)**/
```

```
end:
frees();
}/** figs **/



/* horn.c                                                              */
/* by Morio Kikuchi 2017.1.1                                           */
/* WINDOW SYSTEM:Windows                                               */
/* COMPILER:Visual C++ 4.0                                             */
/*          Open Watcom C/C++ 1.4                                      */
/* COMMANDLINE:cl /w /Fehorn horn.c /link user32.lib gdi32.lib         */
/* COMMANDLINE:wcc386 -w -j horn.c                                     */
/*             link386 -out:horn horn.obj                              */
/*             (wlink name horn file horn)                             */

#include <windows.h>

/*#define _45min*/

#define GKS GetKeyState

#define VGACOLORS 16
#define SF1 0
#define SF2 1

#define CoLUMN /*100*/73
#define RoW /*1638*/23              /* <= 1638 */
#define UdX 10
#define UdY 20

#ifdef _45min
#define HS (45*60)                  /* half span[sec] */
#define To (1*60)                   /* time-out[sec] from among black indication */
#define To_ (1*60)                  /* time-out[sec] from among green indication */
#define BrK (1*60)                  /* break[sec] (Shift+B => green indication) */
#define BrK_ (30)                   /* break[sec] (Shift+M => green indication) */
#define _1stHORN 10                 /* 1st horn[sec] (interval till 2nd horn) */
#define X 2
#define Y 1
#else
#define HS (10)                     /* half span[sec] */
#define To (6)                      /* time-out[sec] from among black indication */
#define To_ (6)                     /* time-out[sec] from among green indication */
#define BrK (6)                     /* break[sec] (Shift+B => green indication) */
#define BrK_ (4)                    /* break[sec] (Shift+M => green indication) */
```

```
#define _1stHORN 3                  /* 1st horn[sec] (interval till 2nd horn) */
#define X 2
#define Y 1
#endif

char charflag=1;
int XRESO,YRESO,YrESO,WB,COLUMN,ROW_V,UDX,UDY,DX_FRAME,DY_FRAME,DY_CAPTION;
int refill,function,c_trans,pflag=1,callflag,addflag,Sflag,TO,BRK;
long s_trans,msec_g,base_g,dtime2,dtime01,total,xg,yg,zg;

void (*pp[1])();

typedef struct {
unsigned char red,green,blue;} srgb;
srgb irgb[VGACOLORS];
typedef struct {
unsigned long back_;int back,fore;} bf;
bf bfset[]={{WHITENESS,15,0},{BLACKNESS,0,15}};

HINSTANCE hinstance;
HWND hwnd;
HDC hdcdisplay,hdctmp1;
HBITMAP hbitmap1;
HFONT hfont;
HBRUSH hbrush;

void figs(void),call(int),closegraph_(),setstccolor(int),restore_in_PAINT(void),
     bitblt(char,int,int,int,int,int,int),cleardevice_(char,int,int,int,int),
     paint(char,int,int,int,int,int),SS_1(void),printf_(int,double,double),
     total_pm(int);

int mainloop(long,long,int);


int main(int argc,unsigned char **argv)
{
xg=GetTickCount();

hinstance=GetModuleHandle(NULL);  /* main() */
/*hinstance=hinstance_;*/         /* WinMain */

WB=0;
refill=1;

if(access("horn_cp.exe",0)==-1){
printf(" horn_cp.exe(<= horn_cp.c) is required.\n");return 1;
```

```
}
if(initgraph_()==1) return 1;

cleardevice_(1,0,0,XRESO,YRESO);

SS_1();

closegraph_();

return 0;
}/** main **/


void beep_cp(int flag)
{
char buf[10],str[10];
STARTUPINFO si;
PROCESS_INFORMATION pi;

itoa(flag,buf,10);
strcpy(str," ");
strcat(str,buf);

GetStartupInfo(&si);
CreateProcess("horn_cp.exe",str,NULL,NULL,FALSE,0,NULL,NULL,&si,&pi);
}/** beep_cp **/


void keydowns_f2(void)
{
if(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0) charflag=0;  /* in paging() */
}/** keydowns_f2 **/


void kbhit_(void)
{
MSG msg;

if(PeekMessage(&msg,NULL,0,0,PM_REMOVE)){
TranslateMessage(&msg);
DispatchMessage(&msg);
}
}/** kbhit_ */


LRESULT CALLBACK  wndproc_by_kbhit_(HWND hwnd,UINT umsg,WPARAM wparam,LPARAM lparam)
```

```
{
if(wndproc_filer(hwnd,umsg,wparam,lparam)!=0) return 1;

return DefWindowProc(hwnd,umsg,wparam,lparam);
}/** wndproc_by_kbhit_ **/


int wndproc_filer(HWND hwnd,UINT umsg,WPARAM wparam,LPARAM lparam)
{
if(umsg==WM_KEYDOWN){
if(function==2){
keydowns_f2();                        /* only Esc, Pause => end of this program */
return 1;
}

/* except paging() -> */
     if(/*GKS(VK_SHIFT)<0 && */(GKS(VK_ESCAPE)<0 || GKS(VK_PAUSE)<0)){
     refill=0;printf(" Esc");}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('C')<0) {call(0);callflag=10;}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('B')<0) {call(1);callflag=10;}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('M')<0) {call(2);callflag=10;}
else if(callflag==1 && addflag==0 && GKS(VK_SHIFT)<0 && GKS('C')<0)
     {addflag=1;printf(" Shift+C in break : %ld\n",base_g+dtime2+dtime01);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('1')<0)
     {if(Sflag==0) total_pm(1);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('2')<0)
     {if(Sflag==0) total_pm(2);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('3')<0)
     {if(Sflag==0) total_pm(3);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('4')<0)
     {if(Sflag==0) total_pm(4);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('7')<0)
     {if(Sflag==0) total_pm(7);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('8')<0)
     {if(Sflag==0) total_pm(8);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('9')<0)
     {if(Sflag==0) total_pm(9);}
else if(callflag==2 && GKS(VK_SHIFT)<0 && GKS('0')<0)
     {if(Sflag==0) total_pm(0);}
else if(callflag==2 && Sflag==1) Sflag=0;
else return 0;
/* <- except paging() */

return 1;
}/**else if(umsg)**/
else if(umsg==WM_CLOSE){
```

```
if(function==2) charflag=0;         /* in paging() */
else refill=0;                      /* in figs() */

return 1;
}/**else if(umsg)**/
else if(umsg==WM_PAINT){
restore_in_PAINT();

return 1;
}/**else if(umsg)**/

return 0;
}/** wndproc_filer **/


void initsysfont(int type,double ds_)
{
if(type==0){                        /* small */
UDX=UdX;UDY=UdY;

hfont=CreateFont(UdY,UdX,0,0,
                 FW_NORMAL,0,0,0,
                 DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,
                 CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,
                 FIXED_PITCH | FF_ROMAN/*FF_MODERN*/,NULL);
SelectObject(hdcdisplay,hfont);
SelectObject(hdctmp1,hfont);
}
else if(type==1){                   /* large */
UDX=UdX;UDY=UdY;

/*999*/
hfont=CreateFont(UdY*13,UdX*13,0,0,
                 FW_NORMAL,0,0,0,
                 DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,
                 CLIP_DEFAULT_PRECIS,DEFAULT_QUALITY,
                 FIXED_PITCH | /*FF_ROMAN*/FF_MODERN,NULL);
SelectObject(hdcdisplay,hfont);
SelectObject(hdctmp1,hfont);
}
}/** initsysfont **/


void SystemFont(char flag,int x,int y,unsigned char *str,int size)
{
if(flag==0)
```

```
TextOut(hdcdisplay,x,y,str,size);
else if(flag==1)
TextOut(hdctmp1,x,y,str,size);
}/** SystemFont **/


void sentence(char flag,int x,int y,unsigned char *str,int size)
{
SystemFont(flag,x,y,str,size);
}/** sentence **/


void paging(void)
{
int function_old;

function_old=function;function=2;

while(1){
kbhit_();
if(charflag==0) break;
}

function=function_old;
}/** paging **/


void SS_1(void)
{
/*page_title();
BitBlt_full();*/

pp[0]=/*page_1*/figs;
(*pp[0])();

/*SS_2();*/
paging();
}/** SS_1 **/


COLORREF PALETTE(int color)
{
return RGB(irgb[color].red,irgb[color].green,irgb[color].blue);
}/** PALETTE **/
```

```
void printf_(int val,double i,double j)
{
int dx,dy,color;
int columns,mpl;
char buf[11],strm[5],strs[5],bufm[5],bufs[5];

/*999*/
columns=/*7*/69;
mpl=12;

dx=i*UDX;dy=j*UDY;

if(pflag==1) color=7;else {pflag=1;color=15;}
paint(1,dx,dy,UDX*columns,UDY*mpl,color);

itoa(val/60,buf,10);
strcpy(strm,buf);
itoa(val%60,buf,10);
strcpy(strs,buf);
if(strlen(strm)<2) {strcpy(bufm,"0");strcat(bufm,strm);}else {strcpy(bufm,strm);}
if(strlen(strs)<2) {strcpy(bufs,"0");strcat(bufs,strs);}else {strcpy(bufs,strs);}
strcpy(buf,bufm);strcat(buf,":");strcat(buf,bufs);

setstccolor(/*bfset[WB].fore*/c_trans);
sentence(1,dx+UDX*2,dy,buf,strlen(buf));

bitblt(1,dx,dy,UDX*columns,UDY*mpl,dx,dy);
}/** printf_ **/


void paint(char flag,int x,int y,int xsize,int ysize,int color)
{
hbrush=CreateSolidBrush(PALETTE(color));

if(flag==0){
SelectObject(hdcdisplay,hbrush);
PatBlt(hdcdisplay,x,y,xsize,ysize,PATCOPY);
}
else if(flag==1){
SelectObject(hdctmp1,hbrush);
PatBlt(hdctmp1,x,y,xsize,ysize,PATCOPY);
}

DeleteObject(hbrush);
}/** paint **/
```

```
void cleardevice_(char flag,int x,int y,int xsize,int ysize)
{
if(flag==0)
PatBlt(hdcdisplay,x,y,xsize,ysize,bfset[WB].back_);
else if(flag==1)
PatBlt(hdctmp1,x,y,xsize,ysize,bfset[WB].back_);
}/** cleardevice_ **/


void bitblt(char flag,int x,int y,int xsize,int ysize,int x_,int y_)
{
if(flag==1)
BitBlt(hdcdisplay,x_,y_,xsize,ysize,
       hdctmp1,x,y,SRCCOPY);
}/** bitblt **/


void restore_in_PAINT(void)
{
ValidateRect(hwnd,NULL);

bitblt(1,0,0,XRESO,YRESO,0,0);
/*printf(" WM_PAINT\n");*/
}/** restore_in_PAINT **/


void initpalette(void)
{
int i;

irgb[0].red=0;irgb[0].green=0;irgb[0].blue=0;

irgb[9].red=0;irgb[9].green=0;irgb[9].blue=255;  /* blue */
irgb[10].red=0;irgb[10].green=255;irgb[10].blue=0;  /* green */
irgb[11].red=0;irgb[11].green=255;irgb[11].blue=255;  /* cyan */
irgb[12].red=255;irgb[12].green=0;irgb[12].blue=0;  /* red */
irgb[13].red=255;irgb[13].green=0;irgb[13].blue=255;  /* magenta */
irgb[14].red=255;irgb[14].green=255;irgb[14].blue=0;  /* yellow */

irgb[15].red=255;irgb[15].green=255;irgb[15].blue=255;

for(i=1;i<7;i++){                    /* 1 -> 6 */
if(irgb[9+(i-1)].red==255)
irgb[i].red=127+64;
if(irgb[9+(i-1)].green==255)
```

```
irgb[i].green=127+64;
if(irgb[9+(i-1)].blue==255)
irgb[i].blue=127+64;
}

for(i=7;i<9;i++){                    /* 7, 8 */
irgb[i].red=127+/*64*/16*(8-i);
irgb[i].green=irgb[i].red;
irgb[i].blue=irgb[i].red;
}
}/** initpalette **/


void setstccolor(int color)
{
SetTextColor(hdcdisplay,PALETTE(color));
SetTextColor(hdctmp1,PALETTE(color));
}/** setstccolor **/


int mainloop(long base,long millisecond,int color)
{
int flag,size,beepflag=0,onceflag=0;
long oldtime,nowtime,dtime/*,millisecond*/;
long sec_new,sec_old;

/*if(callflag==2) {dtime2=0;dtime01=0;}else dtime01=0;*/
if(color==0) flag=0;else flag=1;
size=millisecond/1000;               /* second */

c_trans=color;
sec_old=base/1000;
if(flag==0)
printf_(sec_old,X,Y);                /* second */
else
printf_(size,X,Y);

oldtime=xg;                          /* <= figs(), call() */

while(1){
kbhit_();
if(refill==0){                       /* Esc */
zg=base_g+dtime2+dtime01;
if(callflag==0 || (callflag==1 && addflag==0)){  /* in c, b */
before:
size=0;
```

```
if((yg=zg-msec_g)>0) callflag=4;
else callflag=3;
if(callflag==0)
printf(" callflag=%d : %ld\n",callflag,zg);
else
printf(" callflag=%d addflag=%d : %ld\n",callflag,addflag,zg);
}/**if(callflag,addflag)**/
else if(callflag==1 && addflag==-1){  /* in c from among b */
millisecond-=TO*1000;size-=TO;
if(dtime01<=BRK*1000) {addflag=0;goto before;}
else{
if((yg=zg-(msec_g+millisecond))>0) callflag=6;
else callflag=5;
}
printf(" callflag=%d addflag=%d : %ld\n",callflag,addflag,zg);
}/**else if(callflag,addflag)**/
else{
printf(" callflag=%d : %ld\n",callflag,zg);
}/**else(callflag,addflag)**/

break;
}/**if(refill)**/

if(callflag==10) break;              /* => figs' mainloop() */
if(addflag==1) {millisecond+=TO*1000;size+=TO;beepflag=0;addflag=-1;}
if(callflag==1 && addflag==-1 && onceflag==1 && dtime01>BRK*1000) onceflag=0;

if(1){
if((callflag==0 || (callflag==1 && addflag==0)) && onceflag==0){  /* in c, b */
before_:
if((zg=base_g+dtime2+dtime01)-msec_g>0){
printf(" temporary half span ended(1st Shift+) : %ld\n",zg);
onceflag=1;
}
}
else if(callflag==1 && addflag==-1 && onceflag==0){  /* in c from among b */
if(dtime01<=BRK*1000) {/*addflag=0;*/goto before_;}
else{
if((zg=base_g+dtime2+dtime01)-(msec_g+(millisecond-TO*1000))>0){  /* ==small */
printf(" temporary half span ended(2nd Shift+) : %ld\n",zg);
onceflag=2;
}
}
}/**else if(callflag,addflag)**/
}/**if(1)*/
```

```
nowtime=GetTickCount();
dtime=nowtime-oldtime;

if(callflag==2){
if(base+dtime>=msec_g) {beep_cp(2);dtime2=dtime;refill=0;/*break;*/goto last;}
dtime2=dtime;
}
else{
if(base+dtime>=millisecond) {beep_cp(1);dtime01=dtime;/*break;*/goto last;}
dtime01=dtime;
}

if(1){
if(callflag==1 && dtime>BRK*1000) c_trans=9;
else c_trans=color;

if(flag==0){
sec_new=(base+dtime)/1000;
if(sec_new>sec_old) printf_(/*size-*/sec_new,X,Y);
}
else{
sec_new=(base+dtime)/1000;
if(sec_new>sec_old) printf_(size-sec_new,X,Y);
}

sec_old=sec_new;

if(beepflag==0 && callflag<=1 && dtime>=millisecond-_1stHORN*1000){
/*beep(HORN_S);delay_(HORN_D);beep(HORN_S);*/beep_cp(0);
beepflag=1;
}
}
else{
last:

if(callflag==1 && addflag==-1 && dtime>BRK*1000) c_trans=9;
else c_trans=color;

if(flag==0){
sec_new=(base+dtime)/1000;
if(sec_new>sec_old) printf_(/*size-*/sec_new,X,Y);
}
else{
sec_new=(base+dtime)/1000;
if(sec_new>sec_old) printf_(size-sec_new,X,Y);
}
```

```
sec_old=sec_new;

break;
}
}

addflag=0;

return size;
}/** mainloop **/


void setup(void)
{
DX_FRAME=GetSystemMetrics(SM_CXSIZEFRAME);
DY_FRAME=GetSystemMetrics(SM_CYSIZEFRAME);
DY_CAPTION=GetSystemMetrics(SM_CYCAPTION);

ROW_V=RoW;

XRESO=CoLUMN*UdX;
YrESO=UdY*RoW;
YRESO=YrESO;
}/** setup **/


int initgraph_(void)
{
WNDCLASS wndclass;

initpalette();
setup();

wndclass.hInstance    =hinstance;
wndclass.lpszClassName="SSCLASS";
wndclass.lpszMenuName =NULL;
wndclass.lpfnWndProc  =(WNDPROC)wndproc_by_kbhit_;
wndclass.style        =CS_HREDRAW | CS_VREDRAW;
wndclass.hIcon        =LoadIcon(hinstance,"MYICON");
wndclass.hCursor      =LoadCursor(NULL,IDC_ARROW);
wndclass.cbClsExtra   =0;
wndclass.cbWndExtra   =0;
if(WB==0)
wndclass.hbrBackground=GetStockObject(WHITE_BRUSH);
else
```

```
wndclass.hbrBackground=GetStockObject(BLACK_BRUSH);

if(RegisterClass(&wndclass)==0) return 1;

hwnd=CreateWindow("SSCLASS","SS",
                  WS_OVERLAPPEDWINDOW,
                  275,0,XRESO+DX_FRAME*2,YRESO+DY_CAPTION+DY_FRAME*2,
                  NULL,NULL,hinstance,NULL);
if(hwnd==NULL) {MessageBox(NULL,"Memory not enough.","SS",MB_OK);return 1;}

SetWindowPos(hwnd,HWND_TOP,0,0,0,0,SWP_NOMOVE | SWP_NOSIZE);
ShowWindow(hwnd,SW_SHOWDEFAULT);

hdcdisplay=GetDC(hwnd);

hdctmp1=CreateCompatibleDC(hdcdisplay);
hbitmap1=CreateCompatibleBitmap(hdcdisplay,XRESO,YRESO);  /* visual page */
SelectObject(hdctmp1,hbitmap1);

SetBkMode(hdcdisplay,TRANSPARENT);
SetBkMode(hdctmp1,TRANSPARENT);

SetBkColor(hdcdisplay,PALETTE(bfset[WB].back));
SetBkColor(hdctmp1,PALETTE(bfset[WB].back));

return 0;
}/** initgraph_ **/


void closegraph_(void)
{
DeleteObject(hfont);
DeleteDC(hdctmp1);
DeleteObject(hbitmap1);

ReleaseDC(hwnd,hdcdisplay);
DestroyWindow(hwnd);

UnregisterClass("SSCLASS",hinstance);
}/** closegraph_ **/


void total_pm(int pm)
{
int dlt;
long dlt_;
```

```
Sflag=1;

if(pm==1)      {printf(" Shift+1");dlt=To;}
else if(pm==2) {printf(" Shift+2");dlt=To_;}
else if(pm==3) {printf(" Shift+3");dlt=BrK;}
else if(pm==4) {printf(" Shift+4");dlt=BrK_;}
else if(pm==7) {if(total>=To)   {printf(" Shift+7");dlt=-To;}
                else            {printf(" Shift+7 : invalid(total)\n");return;}}
else if(pm==8) {if(total>=To_)  {printf(" Shift+8");dlt=-To_;}
                else            {printf(" Shift+8 : invalid(total)\n");return;}}
else if(pm==9) {if(total>=BrK)  {printf(" Shift+9");dlt=-BrK;}
                else            {printf(" Shift+9 : invalid(total)\n");return;}}
else if(pm==0) {if(total>=BrK_) {printf(" Shift+0");dlt=-BrK_;}
                else            {printf(" Shift+0 : invalid(total)\n");return;}}

/*if(base_g+dtime2>msec_g+dlt*1000) {printf(" : invalid(msec_g)\n");return;}*/

total+=dlt;
msec_g+=dlt*1000;

if(dlt>0){
printf(" : Δ=%dmin%dsec  total=%dmin%dsec\n",dlt/60,dlt%60,total/60,total%60);
}
else{
dlt*=-1;

if((dlt_=base_g+dtime2-msec_g)<=0){
printf(" : Δ=-%dmin%dsec  total=%dmin%dsec\n",dlt/60,dlt%60,total/60,total%60);
}
else{
printf(" : Δ=-%dmin%dsec  total=%dmin%dsec",dlt/60,dlt%60,total/60,total%60);
printf(" (already ended : %ld before)\n",dlt_);
pflag=2;
printf_(0,X,Y);
}
}
}/** total_pm **/


void call(int flag)
{
int size;

xg=GetTickCount();
dtime01=0;
```

```
if(flag==0){
callflag=0;TO=To;          size=TO; printf(" Shift+C : %ld\n",base_g+dtime2);
}
else if(flag==1){
callflag=1;BRK=BrK;TO=To_; size=BRK;printf(" Shift+B : %ld\n",base_g+dtime2);
}
else if(flag==2){
callflag=1;BRK=BrK_;TO=To_;size=BRK;printf(" Shift+M : %ld\n",base_g+dtime2);
}

size=mainloop(0,size*1000,9+callflag);
/*after_mainloop();*/refill=1;

s_trans=size*1000;
total+=size;

/*if(callflag<=1 && size>0) beep(HORN_M);
else */if(callflag!=4 && callflag!=6){
printf(" Δ=%dmin%dsec  total=%dmin%dsec\n",size/60,size%60,total/60,total%60);
}
else{
printf(" Δ=%dmin%dsec  total=%dmin%dsec",size/60,size%60,total/60,total%60);
printf(" (already ended : %ld before)\n",yg);
pflag=2;
printf_(0,X,Y);
}
}/** call **/


void figs(void)
{
int i,j;
long oldtime,nowtime,dtime;
char buf[CoLUMN];

initsysfont(SF1,0);

strcpy(buf,"Key assignment:");
i=2;j=14;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));

strcpy(buf,"Shift+C:time-out from among gap (1 min)");
i+=15;j++;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));
```

```
strcpy(buf,"Shift+B:break (1 min)");
j++;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));

strcpy(buf,"Shift+M:break (0.5 min)");
j++;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));

strcpy(buf,"Shift+C in break:time-out from among break (1 min)");
j++;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));

strcpy(buf,"Esc:escape from the loop");
j++;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));

strcpy(buf,"If we describe #define _45min, the above values are used");
i=2;j+=2;
sentence(1,i*UDX,j*UDY,buf,strlen(buf));

printf(" HS:%dsec To:%dsec To_:%dsec BrK:%dsec BrK_:%dsec _1stHORN:%dsec\n\n",
         (int)HS,(int)To,(int)To_,(int)BrK,(int)BrK_,(int)_1stHORN);

initsysfont(SF2,0);

total=0;
base_g=0;
msec_g=HS*1000;
oldtime=xg;

while(1){
callflag=2;
s_trans=0;

dtime2=0;dtime01=0;
mainloop(base_g,msec_g,0);
if(refill==0) break;
msec_g+=s_trans;

nowtime=GetTickCount();
dtime=nowtime-oldtime;

if(dtime<msec_g) {base_g=dtime;xg=nowtime;}  /* msec_g=msec_g */
else break;
}/**while(1)**/
```

```
printf(" half span=%ld[msec] %dmin%dsec\n",msec_g,(msec_g/1000)/60,(msec_g/1000)%60);
dtime=base_g+dtime2+dtime01;
printf(" real span=%ld[msec] %dmin%dsec\n",dtime,(dtime/1000)/60,(dtime/1000)%60);
}/** figs **/



/* horn_cp.c                                                          */
/* by Morio Kikuchi 2017.1.1                                          */
/* WINDOW SYSTEM:Windows                                              */
/* COMPILER:Visual C++ 4.0                                            */
/*          Open Watcom C/C++ 1.4                                     */
/* COMMANDLINE:cl /w /Fehorn_cp horn_cp.c /link user32.lib gdi32.lib  */
/* COMMANDLINE:wcc386 -w -j horn_cp.c                                 */
/*             link386 -out:horn_cp horn_cp.obj                       */
/*             (wlink name horn_cp file horn_cp)                      */
/*             (wlink name horn_cp file horn_cp)                      */

#include <windows.h>

#define HORN_S 100                  /* short horn[msec] */
#define HORN_M 200                  /* medium horn[msec] */
#define HORN_L 400                  /* long horn[msec] */
#define DELAY 200                   /* delay[msec] */

void beep(long),delay_(long);


int main(int argc,unsigned char **argv)
{
int val;

if(argc==1) return 1;

val=atoi(argv[1]);

if(val==0){
beep(HORN_S);
delay_(DELAY);
beep(HORN_S);
}
else if(val==1) beep(HORN_M);
else if(val==2) beep(HORN_L);

return 0;
}/** main **/
```

```
void beep(long millisecond)
{
Beep(888,millisecond);
}/** beep **/


void delay_(long millisecond)
{
long oldtime,nowtime;

oldtime=GetTickCount();

while(1){
/*kbhit_();
if(refill==0) break;*/

nowtime=GetTickCount();
if(nowtime-oldtime>=millisecond) break;
}
}/** delay_ **/
```