

平成 20 年度 冬入試

東京大学情報理工学系研究科創造情報学専攻

## プログラミング

### 注意事項

1. 試験開始の合図まで、この問題冊子を開いてはいけません。
2. この表紙の下部にある受験番号欄に受験番号を記入して下さい。
3. 答案用紙と下書き用紙が 1 枚ずつ配られる。それぞれに受験番号を記入して下さい。問 1 と問 2 は解答用紙に解答を記入する。途中で余白が足りなくなったら、試験監督に申し出下さい。
4. 各受験者に配られた USB メモリに ASCII コードで書かれたファイル request00.txt, request01.txt, ..., request05.txt, request.txt の 7 個が入っている。改行はすべて CR と LF の対で書かれている (CR は carriage return, LF は line feed である)。試験開始前に、USB メモリから上記のファイルを自分の PC にコピーして下さい。ファイルの中身を見て、空白で区切られた数がたくさんの行にわたって入っていることを確認し、PC から手を離して下さい。ファイルにアクセスできない、あるいは中身が ASCII 文字列として読めないなどの場合は試験監督に申し出下さい。USB メモリの中身は全受験者に共通である。
5. プログラミング言語は各自の得意なものを使用して下さい。
6. プログラミング言語のマニュアルは 1 冊に限り試験中に参照してもよい。ネットワーク接続は禁止であるが、各自の PC に入っているライブラリやプログラムを使用・流用することは自由である。
7. 試験終了時まで、自分の PC 上に受験番号名のディレクトリ/フォルダを作成し、作成したプログラムおよび関連ファイルをその下にコピーして下さい。作成したディレクトリ/フォルダを各受験者に渡された USB メモリにコピーして下さい。
8. 試験終了時に、USB メモリ、答案用紙、下書き用紙を回収する。
9. 回収後、試験監督が周回し、各受験者の結果をごく簡単に確認するので、そのまま座席で待機して下さい。全員の確認が終わるまで部屋を出てはいけません。
10. 午後のプログラミングの口頭試問中にプログラムの動作をより精密に確認する。各自の PC 上でプログラムがすぐに実行できるようにしておきなさい。この問題は長文であり、難しいので、口頭試問で、どのように問題を解析し、データ構造を設計したか、さらには例題をどこまでできたかを答えられることが重要である。
11. 全員の確認が終了した後、各自の PC とこの問題冊子を残し、部屋から退出して下さい。

受験番号 \_\_\_\_\_

コンピュータで集中制御しているエレベータシステム (図 1) の離散時刻シミュレーションを以下の条件のもとで行なおう。簡単のため、建物はビル地下なし・地上 10 階とする。

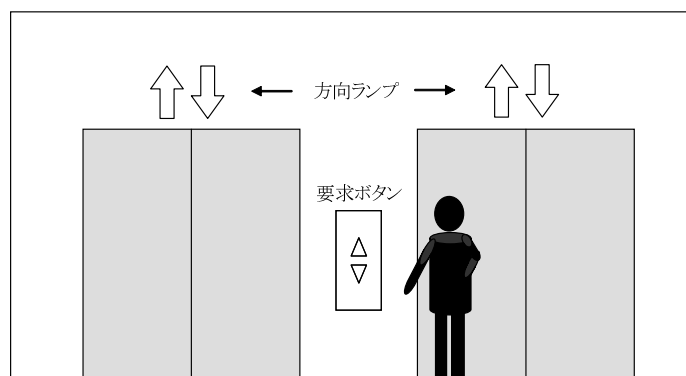


図 1. エレベータシステム

- [E0] 時刻や時間は秒単位で、秒未満の単位はない。ただし、同時刻でも順序は厳密に守る。シミュレーションは時刻 0 から開始する。
- [E1] エレベータに定員はなく、何人でも乗れる。
- [E2] 各階に上向きボタンと下向きボタンがある。乗客はこれを押してサービスを要求する。
- [E3] 各階に各々のエレベータの移動方向を示す方向ランプがある。
- [E4] 時刻 0 では、すべてのエレベータは 1 階で扉を閉じ、方向ランプを消して停止している。
- [E5] 乗客は目的階に着くまで一方向移動することが保証される。乗客は行きたい方向の方向ランプが点灯しているか、消灯して扉が開いているエレベータに乗る。
- [E6] エレベータは移動開始直後あるいは停止直前には階の間を 3 秒で加速移動あるいは減速移動し、その他の場合は階の間を 1 秒で高速移動する。ただし、ある階から出発し、すぐ隣の階で停止する場合は 4 秒かかる。エレベータは階の間で動作を変更できない。図 2 を参照のこと。

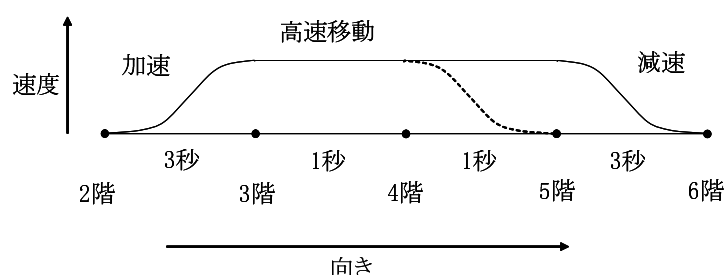
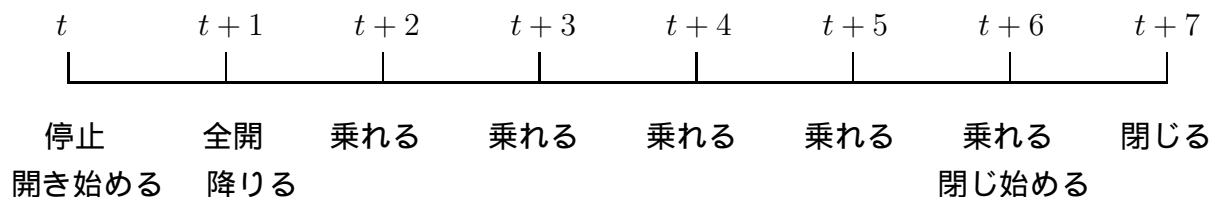


図 2. エレベータの動き。

停止していた 2 階から 6 階へ移動する様子を示している。

- [E7] エレベータが通過する予定の階  $f$  から乗るには、エレベータが  $f$  の 1 つ前の階を通過する時刻または 1 つ前の階から移動し始める時刻を  $t$  とすれば、時刻  $t_b \leq t$  に同じ方向のボタンを押さなければならない。図 2 において、エレベータがちょうど 4 階の位置にあるとき、5 階で上向きボタンを押すと、破線の通り、そこから減速が可能になる。
- [E8] 停止の 1 秒後に扉が完全に開き、降りる人は全員その瞬間に降りる。エレベータ内に開閉ボタンはなく、それから 5 秒間開いたままとなる。

[E9] 下のように、扉全開の5秒後に扉が閉じ始め、1秒で閉まる。扉全開の1秒後 ( $t+2$ ) から扉が閉じ始める  $t+6$  までに来た同じ方向の人は、ボタンを押して、待たずに乗れる。



[E10] ある階  $f$  で上向きボタンを押したとき、 $f$  より下にいて、条件 E7 により  $f$  で停止することが可能なエレベータのどれか1つは通過せずに停止する。下向きの場合も同様。

問1 エレベータが1台として、以下の質問に答えなさい。解答は答案用紙に書くこと。計算過程 (図でもよい) を必ず書くこと。

- (1-1) 時刻 100 にエレベータは3階に扉を閉じて停止していた。Aさんが1階から7階に行くボタンを押した。Aさんが目的階に着くまでの間、建物の中でほかの誰もボタンを押さなかった。Aさんが7階で降りる時刻を求めなさい。
- (1-2) (1-1) と同じ条件のもとで、Aさんがちょうど5階を上向きに通過するときにBさんが6階から10階に行こうとして上向きボタンを押して、6階で乗った (条件 E7 と E10 により、エレベータは6階で停止する)。Aさんが7階で降りる時刻を求めなさい。

シミュレーションプログラムの仕様を詳細化しよう。

乗客の要求は時刻、要求階、目的階の3組で表現する。時刻はシミュレーション開始からの秒数。要求階と目的階は階数で表現する。たとえば、

2008 1 7

は時刻 2008 秒に発生した、1階から7階へ行くという要求を表す。与えられたファイルにはこの形式の要求が1行あたり1件、到着順に書かれている。ただし、ファイルの最初の行は要求がファイルに何個並んでいるかを示す整数である。

- [E11] すべての乗客は乗る前に必ず上向きあるいは下向きのボタンを押す。またすべての乗客は乗った瞬間に目的階のボタンを押す。このため、制御プログラムは任意の時刻の乗客の数を正確に知ることができる。ただし、乗客が乗るまでは目的階を知ることはできない。
- [E12] 同時刻に同じ方向のエレベータの扉が開くと、その方向の乗客は番号の小さいエレベータに乗る。
- [E13] 外で要求ボタンが押されたら、制御コンピュータは、その瞬間に応答すべきエレベータを割り当てる。割り当てられたエレベータは次のE14で示すように停止計画を立てる。ただし、まだ目的階のわからない要求があるので、計画は不完全である。割り当ては要求階に停止するまで変更されない。要求を出した乗客が先 (または同時) に着いた他のエレベータに乗ってしまい、無人の階で扉を開けてしまう可能性がある。

[E14] 割り当てられたエレベータの停止階の計画は、要求が割り当てられた瞬間、または目的階が指定された瞬間、または方向ランプを反転させた瞬間に以下の順で立てる (図3).

- (1) 方向ランプが消灯中なら、要求を最短で満たす向きに方向ランプを点灯する.
- (2) 方向ランプが上向きなら、現在の停止階を含め現在位置から、停止可能階を下から順に登録する. 次に、最上階から最下階へ順に下向きの要求を登録する. 最後に、最下階から上へ、残っている上向きの要求を登録する. この計画での一番上の停止階を  $\Omega$  と呼ぶ. 時刻  $t$  における  $\Omega$  を  $\Omega_t$  と書く.

方向ランプが下向きの場合も同様である ( $\Omega$  は計画での一番下の停止階になる).

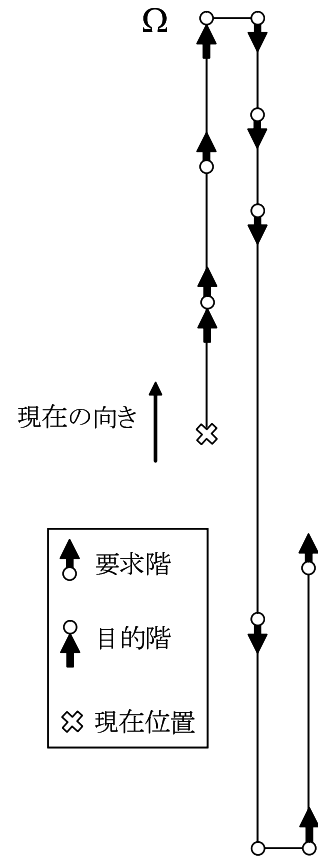


図3. 計画の概念図

問2 エレベータは1台とする. ファイル request00.txt に要求が並んでいる. 以下のそれぞれのタイミングにおける停止計画を図3にならって解答用紙に図示しなさい. 図には階を示す数をつけること.

- (2-1) 7秒後のすべての要求を受け付けた直後.
- (2-2) 10秒後のすべての要求を受け付けた直後.

[E15] エレベータは作成した計画に従って運行する. ただし、以下の細かい規則がある. ここでも方向ランプが上向きの場合だけを記述する.  $U_f$  を階  $f$  における上向き要求と階  $f' > f$  の要求の集合,  $D_f$  を階  $f$  における下向きの要求と階  $f' < f$  における要求の集合とする (要求には目的階の指定を含む).

[方向ランプの消灯と早期反転] 乗客1名以上を乗せ、時刻  $t$  に  $f = \Omega_{t-1}$  に停止したときは  $t+1$  に全員が降りる. もし  $\Omega_{t+1} = f$ , かつ  $U_f = \phi$  (空) かつ  $D_f \neq \phi$  であれば、 $t+1$  にランプを下向きに反転させる.  $U_f = D_f = \phi$  であれば、 $t+1$  にランプを消す.

[計画の例外] 乗客が0の状態で時刻  $t$  に階  $f$  に到着したとき、 $f$  に下向きの要求しかなければ、それより上の階に要求があっても、 $t$  に方向ランプを反転する. これは乗客0で扉を開け、そこに要求があるのになにもサービスしないことを避けるためである.

問3 エレベータは1台とする. ファイル request.txt に要求が並んでいる. E0 ~ E15 の条件・規則にもとづいてエレベータを制御した場合、すべての要求について、要求を出してから乗るまでの待ち時間と、乗ってから目的階で降りるまでにかかった時間を計算し、要求の順に自由な形式で出力しなさい. また、それぞれの平均値 (小数点以下2桁以上) も出力しなさい. 出力ファイル名は ans3.txt とする. ファイル request01.txt ~ request05.txt について計算したサンプル結果を問題冊子の最後につけたので、プログラムのチェックに使いなさい. 口頭試問では、これらの計算結果についても尋ねる.

問4 エレベータは1号機から4号機までの4台とする。問3と同じく、ファイルrequest.txtに要求が並んでいる。E13に関して以下の割り当てポリシーを採用し、問3と同様の計算をなさい。出力ファイル名はans4.txtとする。

簡易予測割り当て: 新しい要求が発生すると、各エレベータは問2で示した計画を立て、それにもとづいて新しい要求階に停止するまでにかかる時間を予測する。わからない目的階があるので、次のような単純な予測法をとる。

- (1) 新たな要求階の停止を現在の計画に仮に入れる。計画がない場合は仮の計画をつくる。
- (2) 停止階に止まりながら、現在の方向の一番遠い階(最上階または最下階)まで行って扉を開け、そこから反対方向の一番遠い階に行き、そこからまた現在の方向に行く。途中で要求階に正しい方向で到着するまでの所要時間を計算する。

予測時間が最小のエレベータに要求を割り当てる。複数ある場合は、最小番号のエレベータを割り当てる。

#### [参考資料]

request01.txt

```
4          2 get-on 0 2
0 1 5      時刻 2 に要求 0 番の乗客が 2 秒待ちで乗った。
0 1 4      2 get-on 1 2
4 1 8      4 get-on 2 0
20 6 10    15 get-off 1 13
           時刻 15 に要求 1 番の乗客が 13 秒間乗ったあと降りた。
           26 get-off 0 24
           38 get-on 3 18
           50 get-off 2 46
           63 get-off 3 25
```

```
request.no= 0 wait-time= 2 onboard-time= 24
request.no= 1 wait-time= 2 onboard-time= 13
request.no= 2 wait-time= 0 onboard-time= 46
request.no= 3 wait-time= 18 onboard-time= 25
average-wait-time= 5.5
average-onboard-time= 27.0
```

request02.txt

```
3          13 get-on 0 9
4 4 8      27 get-off 0 14
20 10 5    41 get-on 1 21
28 9 1     52 get-on 2 24
           66 get-off 1 25
           81 get-off 2 29
```

```
request.no= 0 wait-time= 9 onboard-time= 14
request.no= 1 wait-time= 21 onboard-time= 25
request.no= 2 wait-time= 24 onboard-time= 29
average-wait-time= 18.0
average-onboard-time= 22.667
```

request03.txt

```
3          2 get-on 0 2
0 1 8      19 get-off 0 17
40 2 10    52 get-on 1 12
66 4 10    70 get-off 1 18
           88 get-on 2 22
           104 get-off 2 16
```

```
request.no= 0 wait-time= 2 onboard-time= 17
request.no= 1 wait-time= 12 onboard-time= 18
request.no= 2 wait-time= 22 onboard-time= 16
average-wait-time= 12.0
average-onboard-time= 17.0
```

request04.txt 同じ時刻の要求は方向ランプの状態により, 順序が変わる.

```
7          2 get-on 0 2          202 get-on 4 2
0 1 5      16 get-off 0 14       216 get-off 4 14
17 5 1     17 get-on 1 0        217 get-on 6 1
17 5 10    31 get-off 1 14      232 get-off 6 15
100 10 1   47 get-on 2 30       249 get-on 5 33
200 1 5    62 get-off 2 15      263 get-off 5 14
216 5 1    102 get-on 3 2
216 5 10   121 get-off 3 19
```

```
request.no= 0 wait-time= 2 onboard-time= 14
request.no= 1 wait-time= 0 onboard-time= 14
request.no= 2 wait-time= 30 onboard-time= 15
request.no= 3 wait-time= 2 onboard-time= 19
request.no= 4 wait-time= 2 onboard-time= 14
request.no= 5 wait-time= 33 onboard-time= 14
request.no= 6 wait-time= 1 onboard-time= 15
average-wait-time= 10.0
average-onboard-time= 15.0
```

request05.txt 方向ランプの早期反転と計画の例外

```
4          2 get-on 0 2          127 get-off 2 15
0 1 5      16 get-off 0 14       146 get-on 3 36
0 5 1      17 get-on 1 17        163 get-off 3 17
100 7 2    31 get-off 1 14
110 9 2    112 get-on 2 12
```

```
request.no= 0 wait-time= 2 onboard-time= 14
request.no= 1 wait-time= 17 onboard-time= 14
request.no= 2 wait-time= 12 onboard-time= 15
request.no= 3 wait-time= 36 onboard-time= 17
average-wait-time= 16.75
average-onboard-time= 15.0
```

2008 Winter Entrance Examination

Department of Creative Informatics  
Graduate School of Information Science and Technology  
The University of Tokyo

# Programming

## INSTRUCTIONS

1. Do not open this problem brochure until the signal to begin is given.
2. Write your examinee ID below on this cover.
3. An answer sheet and a draft sheet are delivered. Write down your examinee ID on both sheets. Q1 and Q2 should be answered on the answer sheet. If you need another sheet in the examination, raise your hand and request it to the test proctor.
4. The USB memory delivered beforehand to each examinee contains seven ASCII text files: `request00.txt`, `request01.txt`, ..., `request05.txt`, and `request.txt`. Newline is represented by a pair of carriage return (CR) and linefeed (LF) in these files.  
**Before examination starts**, copy these files to your PC and browse them. Make sure you can see a set of numbers separated by blank or newline in these files, and then keep your hands away from your PC. If you cannot read the file properly, consult the test proctor. The contents of the USB memory is common to all examinees.
5. You may choose your favorite programming language.
6. You may consult only one printed manual of the programming language in the examination. **You can use or copy any libraries or programs existing in your PC.**
7. By the end of the examination, make a directory/folder on your PC, whose name is the same as your examinee ID, and put your program files and related files under the directory/folder. Copy the directory/folder into the USB memory.
8. At the end of the examination, the USB memory, the answer sheet and draft sheet are collected.
9. After the collection, stay at your seat, until all examinee results have been checked briefly by the test proctor.
10. After the brief check, try to retain your program execution environment on the PC so as to be able to resume it as soon as possible at the oral examination. **Because the problem is longer than usual, and may be a little so difficult that Q&A in the oral examination will be important, which asks you how you analyzed the problems, what data structure you designed, and how far you can run your program correctly for the sample data files.**
11. **Leave your PC and this brochure together in the room** for the oral examination and leave the room until you are called.

Examinee ID \_\_\_\_\_

Write a discrete event simulation program for an elevator system controlled by a computer (Figure 1). We assume that the building has 10 floors and no basement floors.

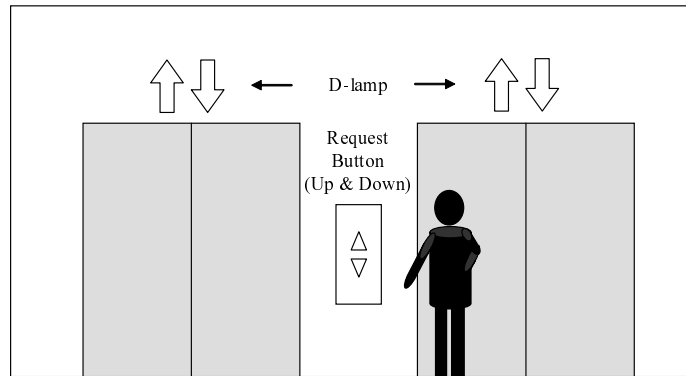


Fig 1. An elevator system.

- [E0]** The time unit is second, and there is no sub-second time quantity. The order of events even at the same time should be strictly obeyed. Simulation will start at the time 0.
- [E1]** Every elevator accommodates unlimited number of passengers.
- [E2]** There are *up*-button and *down*-button at every floor by which passengers can request the elevator service.
- [E3]** On every floor, each elevator has a D-lamp that indicates its direction, “up” or “down”.
- [E4]** At the time 0, all elevators stay at the 1st floor with the doors closed and the D-lamps turned off.
- [E5]** Passengers inside an elevator are guaranteed to move in one direction to their destinations. Passengers only get into the elevator whose D-lamp points to the desired direction, or if the D-lamp is turned off and the door is open.
- [E6]** It takes 3 seconds for an elevator to move between two adjacent floors if it starts and accelerates from one floor to pass the other, or if it decelerates at one floor position to stop at the other floor. Otherwise it passes adjacent floors in 1 second at its maximum speed. It takes 4 seconds to start from a floor and immediately stop at the adjacent floor. The elevator cannot change its action between floors. See Figure 2.

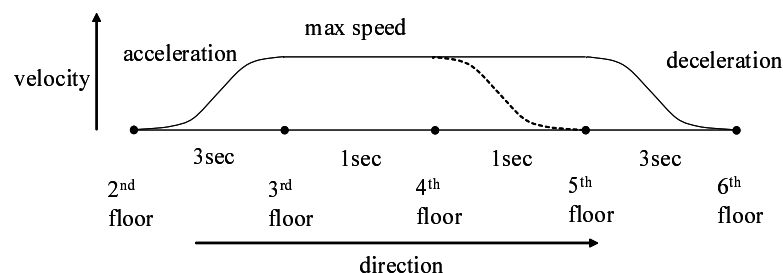


Fig 2. An example of the elevator movement.

The velocity curve of an elevator which moves from the 2nd floor to the 6th floor.

- [E7]** If a passenger on the  $f$ -th floor which will be passed by an elevator which is starting from, or is passing the previous floor at the time  $t$ , he/she must touch the button of the same direction at the time  $t_b \leq t$ . In Figure 2, if a passenger touches an up-button on the 5th floor when the elevator is just passing the 4th floor, the elevator can begin to decelerate as shown by the dashed velocity curve.
- [E8]** In 1 second after an elevator stops at a floor, its door is fully open and all passengers





**Q4** There are four elevators in the system, each numbered from 1 to 4. Requests are in the file `request.txt` as Q3. We use the following assignment policy for E13. Compute the results in the similar way as specified in Q3. Output file should be `ans4.txt`.

**Simple Time Guess Assignment:** When a new request arrives, every elevator is asked to make a plan as shown in Q2, and guess the time to arrive at the source floor. Since there may be some unknown destination floors, we use the following simple procedure.

- (1) Put the new request tentatively into the current plan, or make a tentative new plan if there is not.
- (2) Simulate the elevator movement and compute the time to arrive at the new source floor in the specified direction as follows; stopping at planned floors, move to the furthest floor of the current direction (the uppermost floor or the 1st floor) and open the door, then turn for the opposite furthest floor and open the door there, and finally turn to the current direction, until the new source floor is found.

The elevator which reports the minimum time will be assigned the request. If more than one elevator reports the same minimum, the elevator of the smallest number is assigned.

### [Sample computation]

`request01.txt`

```

4           2 get-on 0 2    ; At the time 2, the passenger of request 0 gets
0 1 5           ; on after waiting 2 seconds.
0 1 4           2 get-on 1 2
4 1 8           4 get-on 2 0
20 6 10        15 get-off 1 13 ; At the time 15, the passenger of request 1 gets
           ; off after 13 seconds riding.
           26 get-off 0 24
           38 get-on 3 18
           50 get-off 2 46
           63 get-off 3 25

```

```

request.no= 0 wait-time= 2 onboard-time= 24
request.no= 1 wait-time= 2 onboard-time= 13
request.no= 2 wait-time= 0 onboard-time= 46
request.no= 3 wait-time= 18 onboard-time= 25
average-wait-time= 5.5
average-onboard-time= 27.0

```

`request02.txt`

```

3           13 get-on 0 9
4 4 8       27 get-off 0 14
20 10 5     41 get-on 1 21
28 9 1      52 get-on 2 24
           66 get-off 1 25
           81 get-off 2 29

```

```

request.no= 0 wait-time= 9 onboard-time= 14
request.no= 1 wait-time= 21 onboard-time= 25
request.no= 2 wait-time= 24 onboard-time= 29
average-wait-time= 18.0
average-onboard-time= 22.667

```

request03.txt

```
3          2 get-on 0 2
0 1 8      19 get-off 0 17
40 2 10    52 get-on 1 12
66 4 10    70 get-off 1 18
           88 get-on 2 22
           104 get-off 2 16
```

```
request.no= 0 wait-time= 2 onboard-time= 17
request.no= 1 wait-time= 12 onboard-time= 18
request.no= 2 wait-time= 22 onboard-time= 16
average-wait-time= 12.0
average-onboard-time= 17.0
```

request04.txt ; Requests at the same time are dealt differently depending on  
; the D-lamp direction.

```
7          2 get-on 0 2          202 get-on 4 2
0 1 5      16 get-off 0 14       216 get-off 4 14
17 5 1     17 get-on 1 0        217 get-on 6 1
17 5 10    31 get-off 1 14      232 get-off 6 15
100 10 1   47 get-on 2 30       249 get-on 5 33
200 1 5    62 get-off 2 15      263 get-off 5 14
216 5 1    102 get-on 3 2
216 5 10   121 get-off 3 19
```

```
request.no= 0 wait-time= 2 onboard-time= 14
request.no= 1 wait-time= 0 onboard-time= 14
request.no= 2 wait-time= 30 onboard-time= 15
request.no= 3 wait-time= 2 onboard-time= 19
request.no= 4 wait-time= 2 onboard-time= 14
request.no= 5 wait-time= 33 onboard-time= 14
request.no= 6 wait-time= 1 onboard-time= 15
average-wait-time= 10.0
average-onboard-time= 15.0
```

request05.txt ; The early D-lamp reversing and plan exception.

```
4          2 get-on 0 2          127 get-off 2 15
0 1 5      16 get-off 0 14       146 get-on 3 36
0 5 1      17 get-on 1 17        163 get-off 3 17
100 7 2    31 get-off 1 14
110 9 2    112 get-on 2 12
```

```
request.no= 0 wait-time= 2 onboard-time= 14
request.no= 1 wait-time= 17 onboard-time= 14
request.no= 2 wait-time= 12 onboard-time= 15
request.no= 3 wait-time= 36 onboard-time= 17
average-wait-time= 16.75
average-onboard-time= 15.0
```