Programming Design, Spring 2016
Lab Exam 1

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

For all the problems in this exam, you are allowed to use any technique.

Problem 1

(30 points) Given n integers and an integer k, please find the kth largest integer among the n integers.

Input/output formats

There are 15 input files. In each file, there are $n + 2$ integers. The first integer is n, the second one is k, and the last n ones are x_1, x_2, ..., and x_n. Two consecutive integers are separated by a white space. It is known that $1 \leq n \leq 200$, $1 \leq k \leq 200$, and $0 \leq x_i \leq 1000$ for all $i = 1, 2, ..., n$. Your program should print out the kth largest value among x_1, x_2, ..., and x_n. If $k > n$, print out -1. For example, the following input

```
4 2 5 6 3 4
```

requires you to print out

```
5
```

Grading criteria

30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.

Problem 2

(20 points) A basketball team has played n games. This team has exactly seven players, and because for each game five players must be the *starting players*, there are $\binom{7}{5} = 21$ possible *starting lineup*. The coach of this team wants to know which starting lineup has the highest winning probability, the number of games won by this lineup, and the number of games lost by this lineup.

Input/output formats

There are 10 input files. In each file, there are $n + 1$ lines. The first line contains a single integer n. It is known that $1 \leq n \leq 200$. Each of the last n lines contains two integers, s_i and r_i, separated by a white space. $s_i \in \{1, 2, ..., 21\}$ is the ID of the starting lineup of game i, and $r_i \in \{0, 1\}$ records the result of game i. If $r_i = 1$, the team won game i; otherwise, the team lost the game.

For example, the following input

```
6
11 1
2 1
2 0
2 1
4 0
4 1
```

records the results of six games. Game 1 was won by lineup 11, game 2 was won by lineup 2, game 3 was lost by lineup 2, ..., and game 6 was won by lineup 4. We may then calculate the winning probabilities of each lineup: 100% for lineup 11, 66.7% for lineup 2, and 50% for lineup 4. You should print out

```
11 1 0
```

to show that lineup 11 has the highest winning probability, it has won 1 game, and it has lost no game. Two consecutive integers should be separated by a white space. If multiple lineups have the same highest winning probability, print out the information for the lineup with the smallest lineup ID. Finally, if a lineup has never been used as a starting lineup, please consider its winning probability as 0.

Grading criteria

20 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.

Problem 3

(20 points) Similar to the team discussed in Problem 2, another basketball team coach wants to do the same thing. However, instead of recording lineup IDs for each game, the lineups are recorded by giving the five starting players a mark that is different from that for the two non-starting players. The coach of this team wants to know which five players for a starting lineup to have the highest winning probability (number of wins divided by the number of games played), the number of games won by this lineup, and the number of games lost by this lineup.

Input/output formats

There are 10 input files. In each file, there are $n+1$ lines. The first line contains a single integer n. It is known that $1 \leq n \leq 200$. Each of the last n lines contains eight integers, $x_{i,1}$, $x_{i,2}$, ..., $x_{i,7}$, and r_i. Two consecutive integers are separated by a white space. $x_{ij} \in \{0,1\}$ records whether player j plays a starting player for game i. If $x_{ij} = 1$, player j is a starting player for game i; otherwise, she/he is not. $r_i \in \{0,1\}$ records the result of game i. If $r_i = 1$, the team won game i; otherwise, the team lost the game. For example, the following input

```
6
0 1 0 1 1 1 1 1
0 1 1 1 1 1 0 1
0 1 1 1 1 1 0 0
0 1 1 1 1 1 0 1
1 1 1 1 1 0 0 0
1 1 1 1 1 0 0 1
```

records the results of six games. The lineup with players 2, 4, 5, 6, and 7 won game 1, that with players 2, 3, 4, 5, and 6 won games 2 and 4 but lost game 3, and that with players 1, 2, 3, 4, and 5 6 won game

6 but lost game 5. We may then calculate the winning probabilities of each lineup: 100% for the first one, 66.7% for the second one, and 50% for the last one. The program should print out

```
2 4 5 6 7 1 0
```

where the first five integers are the player IDs of the highest-winning-probability lineup, the sixth integer is the number of games won by this lineup, and the last integer is the number of games lost by this lineup. Two consecutive integers should be separated by a white space.

If multiple lineups have the same highest winning probability, print out the information for the lineup whose smallest player IDs is the smallest. If more than one lineups have the same the smallest player ID, print out the information for the lineup whose second smallest player IDs is the smallest, etc. For example, if the lineup with players 1, 2, 4, 5, and 7 and that with players 1, 2, 3, 6, 7 have the same winning probability, print out the information for the latter lineup because $3 < 4$. Finally, if a lineup has never been used as a starting lineup, please consider its winning probability as 0.

Grading criteria

20 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.

Problem 4

(30 points) Given an integer x_0 and an integer sequence $(x_1, x_2, ..., x_n)$, we want to replace x_i by x_0 for some $i \in \{1, ..., n\}$ to maximize the length of the longest positive subsequence, which is a subsequence $(x_j, x_{j+1}, ..., x_k)$, $1 \leq j \leq k \leq n$, satisfying $x_i > 0$ for all $i \in \{j, j + 1, ..., k\}$. Note that 0 is not positive. Print out the position to do the replacement and the resulting length of the longest positive subsequence. If there are multiple positions resulting in the same maximum length, print out the one having the smallest index. Even if $x_0 < 0$, we still need to replace one element in the sequence by x_0.

Input/output formats

There are 15 input files. In each file, there are $n + 2$ integers. The first integer is n. The second integer is x_0. The last n integers form the sequence $(x_1, x_2, ..., x_n)$. Two consecutive integers are separated by a white space. It is known that $1 \leq n \leq 100$ and $-100 \leq x_i \leq 100$ for $i \in \{0, 1, ..., n\}$. Your program needs to find the smallest i such that $(x_1, x_2, ..., x_{i-1}, x_0, x_{i+1}, ..., x_n)$, the new sequence formed by replacing x_i by x_0, contains the longest positive subsequence.

For example, the following input

```
7 2 1 -2 3 -4 5 -6 7
```

indicates that $x_0 = 2$ and the sequence is $(1, -2, 3, -4, 5, -6, 7)$. The program should print out

```
2 3
```

because $i = 2$ is the smallest i so that replacing x_i by x_0 results in the longest positive subsequence $(1, 2, 3)$, whose length is 3. Note that it is 2, instead of 4 or 6, that should be chosen as the replacement position. Two consecutive integers should be separated by a white space.

Grading criteria

30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.