CSE214 (Computer Science II)
Homework #2
Linked List, Stack and Queues

Homework guideline:
Solve the problems using linked list, stack and/or queue. Give optimal solution for each problem. Implement your own linked list/stack/queue.

Submission guideline:
Submit the homework through blackboard. Before submission make sure your codes do not have any error, unexecutable code and/or late submission will not receive any credit. Submit your Java code (.java files only) as a single .zip archive. Include a README text file to give the TA's instructions on how to run your code. The .zip file name should be in the following format: < firstname >_< lastname >_< id > _hw< num >.zip
For example, if John Doe with student ID 123456789 is submitting the third homework, the submitted file
should be named john_doe_123456789_hw3.zip

Number Distribution:
(1) 14
(2) 12
(3) 10
(4) 14

Total Marks: 50
Homework posted on: October 2, 2017
Submission Date: October 11, 2017 (11:59 PM)

(1) **Clean up Friend List**

After getting the best student award, Jenny has become very popular in her university. Her Facebook profile is full of friend requests. Being a nice person, she has accepted all the friend requests. Jenny's best friend Phoebe is annoyed as Jenny is giving those newly added friends a lot of time. Phoebe asked Jenny to delete those people from the list as she hardly knows them. To avoid problem in their friendship Jenny decides to remove some of the friends from her friend list. Since she can see the number of mutual friends she uses the following rule to delete some friends while keeping few.

She bucketize the friends based on the number of mutual friends into K buckets, where K is the number of friends she wants to keep (she will keep at most K friends, she might keep less than K). For each bucket, she put the friends in a circle in sorted order based on number of mutual friends (least to most), if multiple friends have same number of mutual friends, she put them in the order as they appear in the input. Counting begins at a friend having least number of mutual friends in the circle and proceeds around the circle in clockwise direction. After K-1 number of friends are skipped, the next person (K^{th}) is deleted. The procedure is repeated with the remaining friends, starting with the next person, going in the same direction and skipping K-1 number of people, until only one friends remains, and she keeps that friend in the list.

How to bucketize? – Find minimum (minMF) and maximum (maxMF) number of mutual friends. Divide the range [minMF, maxMF] into K equal sized buckets. Only the last bucket can have range smaller or larger than the other buckets. Put a friend in a bucket if the number of mutual friends falls in the range of that bucket. For example: [minMF, maxMF] = [2, 11] and K = 3, the range of 3 buckets are [2,4], [5,7], [8,11].

For some test case, there might be 1 or more buckets which are empty. If there are E empty buckets then Jenny keeps K – E friends. If a bucket gets only one friend then Jenny just keeps that friend.

Input Format: Read input from a file "in1.txt". The first line contains T number of test cases. The first line of each test case consists of two space separated integers N and K, denoting the number of newly added friends and the number of friends Jenny wants to keep respectively. The second line consists of N strings representing the names of N newly added friends. The next line consists of integers representing the number of mutual friends of each of the above people with Jenny.

Output Format: Write output in console. Print the names of those K friends whom Jenny wants keep in her friend list in the order as they appear in the input.

Sample Input:

```
1
10 3
a b c d e f g h i j
10 2 4 5 7 3 2 9 11 6
```

Sample Output:

```
a b j
```

(2) **Team Selection**

Coach Sean Miller is looking for an optimal way for his basketball team selection. Initially he has selected M number of players (p_1, p_2, p_3, p_4, …, p_m) with height (h_1, h_2, h_3, h_4, …, h_m). He needs to shortlist and remove some of the players before final selection. So, he called all the players and told them to stand side by side in a single line. Now he started removing a player if the player p_x has someone p_y in his right who is taller than him ($h_x < h_y$).

Write a program for coach Sean Miller to help him eliminate the players under his chosen criteria using Linked List. As he is in hurry, you must make sure the solution is optimal.

Input Format:

Read input from a file "in2.txt". First line contains number of test cases. Second line contains the jersey number of the players. Third line contains heights of the players in centimeters.

Output Format:

Write output in console. Print jersey numbers of the remaining players.

Sample Input:

```
1
1 2 3 4 5 6 7
177 175 165 172 161 170 164
```

Sample Output:

```
1 2 4 6 7
```

(3) **LeakyStack**

Stacks are often used to provide “undo” support in applications like a Web browser or text editor. While support for undo can be implemented with an unbounded stack, many applications provide only *limited* support for such an undo history, with a fixed-capacity stack. When push is invoked with the stack at full capacity, rather than throwing an exception, a more typical semantic is to accept the pushed element at the top while “leaking” the oldest element from the bottom of the stack to make room. Give an implementation of such a LeakyStack abstraction, using a circular array.

Input Format: Read input from a file “in3.txt”. The first line contains T number of test cases. The first line of each test case is the capacity of the LeakyStack. Second line contains the consecutive operations in the browser.

Output Format: Write output in console. Print the content of the stack.

Sample Input:
```
1
3
op_one op_two op_three op_four op_five
```

Sample Output:
```
op_five op_four op_three
```

(4) **John vs The Night King**

Daenerys says, “bring one from the army of dead, to prove their existence”. John Snow and Jorah Mormont head to the north of the wall. There they meet the Night King. They are now in serious trouble. The only way to escape death is to answer a question by the Night King.

Night King shows them a queue of N white walkers of which only M are to be selected. Each white walker has some power associated with it. There are M iterations on the queue. In each iteration, M white walkers are dequeued (if queue has less than M entries, all of them will be dequeued) and the one with maximum power is selected and remaining whites are enqueued back to the queue (in the order they were dequeued) but their power is decreased by one. If there are multiple white walkers with maximum power in those dequeued white walkers, the one which comes first in the queue is selected. If at any time, power of any white walker becomes 0, it can't be decremented any further, it remains the same.

Now, Night King asks John and Jorah to tell him the positions of all the selected white walkers (positions in the initial given queue) in the order they are selected. As they are scared to death having the Night King so close to them, they can't think of anything. They send a raven to you for their rescue. Help them to get the answer fast and save them from the white walkers.

Input Format:
Read input from a file “in4.txt”. The first line contains T number of test cases. The first line of each test case consists of two space separated integers N and M, denoting the number of white walkers in the queue and the number of white walkers that needs to be selected respectively. The next line consists of an array A, A[i] denoting the power of white walker at position i (0≤i≤N-1).

Output Format:
Write output in console. For each of the M iterations, output the position of the selected white walker in that iteration. Position refers to the index at which the white walker was present in the initial given queue (0 based indexing).

Sample Input:

```
1
6 5
1 2 2 3 4 5
```

Sample Output:

```
4 5 3 0 1
```