COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad

IIT PALAKKAD

CS2130: Data Structures and Algorithms Lab

Lab 4 (Stacks and Queues Using Arrays)

1 September, 2018

Time: 3 hrs

General Instructions

- You are expected to follow good programming practices (refer to the book titled *C How to Program* by Dietel and Dietel). Your programs should handle all kinds of test cases - positive, negative, corner cases/boundary conditions, good input, bad input and are never expected to crash on any input.
- You are also expected to provided a high level description of the functions used in your programs. There is no restriction on the format except that it should be concise and precise. Further, the running time and additional space (excluding the space for the arguments) used by the functions should be clearly stated and justified.
- **Please note that poor programming practices and improper documentation can attract a penalty of up to 20% of the marks allocated to a question.**

1 Part 1: Review of Heaps

1. Implement the **insertKey**, **print**, **increseKey** and **extractMax** operations of the max-heap data structure. The skeleton of the code and the input-output format are as specified in the previous assignment. Use the filename **11.c** for submission. [30]

   Input-Output Format: The first line contains the value of n, the maximum size of the array $\boldsymbol{A}$ representing the heap. The subsequent lines will contain specific operations on the heap. The input-output format are as follows

   INS x denotes **insertKey**$(\boldsymbol{A}, x)$. Insert the element key into the heap represented by $\boldsymbol{A}$. Output **-1** if the operation was unsuccessful.
   EXT denotes **extractMax**$(\boldsymbol{A})$. Delete the root of the max heap represented by $\boldsymbol{A}$. Output the deleted element if the operation is successful and -1 otherwise.
   PRT denotes **print**$(\boldsymbol{A})$. Output should be the elements of the heap in the increasing order of their indices in the underlying array, one element per line.
   INC i x denotes **increaseKey**$(\boldsymbol{A}$**, i, x)**. Increases the value of $\boldsymbol{A}[i]$ to $newVal$. Output **-1** if the operation was unsuccessful.

2 Part 2: Implementing Stacks and Queues using Arrays

1. Write code to implement the Stack ADT using arrays. The elements in the stack should have non-negative integral data.

   Input-Output Format: The first line contains the value of n, the maximum size of the underlying array. The subsequent lines will contain specific operations on the stack.

Continued on next page. . .

PSH x denotes **push(.,.)**. Insert the element x into the stack. Output -1 if the operation was unsuccessful.
POP denotes **pop(.)**. Delete the topmost element from the stack. Output the value of the deleted element if the operation was successful and -1 otherwise.
TOP denotes **peek(.,.)**. Obtain the value of the topmost element from the stack. Return the value of this element if the operation is successful and -1 otherwise.
PRT denotes **print(.)**. Output the elements of stack in the reverse order in which they were inserted, one element per line.
SZE denotes **size(.)**. Output the number of elements in the stack.
EMP denotes **isEmpty(.)**. Return 1 if the stack is empty and 0 otherwise.
FUL denotes **isFull(.)**. Return 1 if the stack is full and 0 otherwise.

```
Sample Input:
10
PSH 5
TOP
SZE
PSH 4
PRT
POP
POP
EMP

Expected Output:
5
1
4
5
4
5
1
```

2. Write code to implement the Circular Queue ADT using arrays. A skeleton of the code is given and you should implement the functions that are left empty. Note that if the maximum size of the underlying array is N, then the queue should be allowed to hold only a maximum of $N-1$ elements. Further, the queue should be capable of containing $N-1$ elements irrespective of the relative positions of front and rear.

   Input-Output Format: The input will be given in a file with the following format. The first line contains the value of n, the maximum size of the underlying array. The subsequent lines will contain specific operations on the queue.
   ENQ x denotes **enqueue(.,.)**. Insert the element x into the queue. Output -1 if the operation was unsuccessful.
   DEQ denotes **dequeue(.)**. Delete the first element from the queue. Output the value of the deleted element if the operation is successful and -1 otherwise.
   FRN denotes **peekFront(.,.)**. Obtain the value of the first element from the queue. Output the value of this element if the operation is successful and -1 otherwise.

Question 2 continues on the next page. . .

PRT denotes **print(.)**. Display the elements of queue in the order in which they were inserted, one element per line.
SZE denotes **size(.)**. Output the number of elements in the queue.
EMP denotes **isEmpty(.)**. Return 1 if the queue is empty and 0 otherwise.
FUL denotes **isFull(.)**. Return 1 if the queue is full and 0 otherwise.

```
Sample Input:
10
ENQ 5
FRN
SZE
ENQ 4
PRT
DEQ
DEQ
EMP

Expected Output:
5
1
5
4
5
4
1
```