National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B.Tech.(CSE)
CS2092D Programming Laboratory
Assignment #5: Part-2

Submission deadline (on or before):

- 27.10.2021, 9:00 AM

Policies for Submission and Evaluation:

- You must submit your assignment (**both parts, Part-1 and Part-2 together**) in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors using gcc compiler.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip`

  (Example: *ASSG5_BxxyyyyCS_LAXMAN.zip*). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.
- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: *ASSG5_BxxyyyyCS_LAXMAN_3.c*). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found here.

General Instructions

- Programs should be written in C language and compiled using gcc compiler. **Submit the solutions to the questions through the submission link in Eduserver**.
- Check your programs with sufficiently large values of inputs with in the range as specified in the question.
- Global and/or static variables should not be used in your program.

QUESTIONS

3. Write a menu driven program to implement a STACK S of at most n elements using an array $A[0..n-1]$. STACK is to be declared as a struct with an attribute *top* and an array $A[0..n-1]$ as members. The attribute *top* indexes the most recently inserted element in the stack. The stack consists of elements $A[0..S.top]$ where *A[0]* is the element at the bottom of the stack and *A[S.top]* is the element at the top. Your program must contain the following functions: (In function prototypes, S denotes a stack, and k denotes a character. All operations should run in *O(1)* time.)

   - MAIN() - repeatedly reads a character '*i*', '*d*', '*p*' or '*t*' from the terminal and calls the appropriate function until character '*t*' is entered.
   - STACK-EMPTY(S) - checks whether the Stack S is empty or not. It returns -1 when S is empty, else returns 1.
   - STACK-FULL(S) - checks whether the Stack S is full or not. It returns -1 when S is full, else returns 1.
   - PUSH(S, k) - inserts k to the top of S (check STACK-FULL() inside this function).
   - POP(S) - prints and deletes the most recently inserted element from S (check STACK-EMPTY() inside this function).
   - PEEK(S) - returns the top element of S.

   Input format:

   - The first line of the input contains an integer $n \in [0, 10^5]$, the capacity of stack S.
   - Upcoming lines starts with a character from '*i*', '*d*', '*p*', or '*t*' followed by zero or one upper case alphabet.
   - Character '*i*' is followed by a character `C` $\in [A..Z]$ separated by a space. In this operation, the character `C` is inserted to S by calling the function PUSH(S, k).
   - Character '*d*' is to perform the `Pop` operation on S by calling the function POP(S).
   - Character '*p*' is to perform the `Peek` operation on S by calling the function PEEK(S).
   - Character '*t*' is to 'terminate' the program.

   Output Format:

   - The output (if any) of each command should be printed on a separate line.
   - For option '*i*': Print `-1` if S is full.
   - For option '*p*'and '*d*': Print `-1` if S is empty.

   Sample Input :

```
5
i D
i H
d
i A
i D
i Z
i K
i X
d
d
d
p
i Y
d
```

```
d
d
p
d
t
```

Sample Output:

```
H
-1
K
Z
D
A
Y
A
D
-1
-1
```

4. Write a menu driven program to implement a QUEUE Q using an array $A[0..n-1]$. Queue is to be declared as a struct with three attributes:- *head, tail* and array $A[0..n-1]$. The attribute *Q.head* indexes its head and *Q.tail* indexes the next location at which a newly arriving element will be inserted into the queue. The elements in the queue reside in locations *Q.head, Q.head+1,...., Q.tail-1*, where the location *0* immediately follows location *n-1* in a circular order.Your program must contain the following functions: (in function prototypes, Q denotes a Queue and S denotes a string of characters. All operations should run in *O(1)* time.)

   - MAIN() - repeatedly reads a character '*i*', '*d*', '*f*' or '*e*' from terminal and calls the appropriate function until character '*t*' is entered.
   - QUEUEFULL(Q) - checks whether the Queue is full or not. It returns -1 when Q is full, else returns 1.
   - QUEUEEMPTY(Q) - checks whether the Queue is empty or not. It returns -1 when Q is empty, else returns 1.
   - ENQUEUE(Q, S) - inserts the element S to the tail of Q (check QUEUEFULL() inside this function).
   - DEQUEUE(Q) - prints and deletes the element from the head of Q (check QUEUEEMPTY() inside this function).

   Input format:

   - The first line of the input contains an integer $n \in [0, 10^5]$, the size of the array A.
   - Upcoming lines starts with a character from '*i*', '*d*', '*f*' or '*e*' followed by zero or one string of characters.
   - Character '*i*' is followed by a string $S \in [A-Z, a-z, 0-9]$ of at most 20 characters separated by a space. In this operation, string S is inserted to the tail of Q by calling the function ENQUEUE(Q, S).
   - Character '*d*' is to perform the `Dequeue` operation on Q by calling the function DEQUEUE(Q).
   - Character '*f*' is to check whether the Queue is full or not by calling the function QUEUEFULL(Q).
   - Character '*e*' is to check whether the Queue is empty or not by calling the function QUEUEEMPTY(Q).
   - Character '*t*' is to 'terminate' the program.

   Output Format:

   - The output (if any) of each command should be printed on a separate line.

- For option 'i'and 'f': Print -1 if Q is full.
- For option 'd'and 'e': Print -1 if Q is empty.
- For option 'f': Print 1 if Q is not full.
- For option 'e', Print 1 if Q is not empty.

Sample Input

```
5
i Abc20
i Xyz38
d
d
d
e
i Nit40
e
i IIT35
i IIM42
f
i Fix33
i Calc27
f
d
t
```

Sample Output

```
Abc20
Xyz38
-1
-1
1
1
-1
Nit40
```