National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B. Tech.(CSE)
CS2092D Programming Laboratory
Assignment #6

Submission deadline (on or before):

- 25.10.2022, 11:55 PM

Policies for Submission and Evaluation:

- You must submit your assignment 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

- The source codes must be named as

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

  (For example: $ASSG1\_BxxyyyyCS\_LAXMAN\_1.c$).

  Create separate files for each question and place all of them in a single folder named

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

  (Example: $ASSG1\_BxxyyyyCS\_LAXMAN$). Then Zip the **folder**.

- Submit the single ZIP (.zip) file you have created(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: $ASSG1\_BxxyyyyCS\_LAXMAN.zip$). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

  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 at: https://minerva.nitc.ac.in/?q=node/650.

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.
- You shouldn't change the function prototype specified in the question.

QUESTIONS

1. 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 structure with an attribute *top* and the 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 an integer. All operations should run in $O(1)$ time.)

    - MAIN() - repeatedly reads a character 'i', 'd', 'e' 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.
    - STACK-FULL(S) - checks whether the Stack S is full or not.
    - PUSH(S, k) - inserts k to the top of S.
    - POP(S) - returns the most recently inserted element from S and deletes it from S.

    Input format:

    - The first line of the input contains an integer $n \in [0, 10^5]$, size of the stack S.
    - Upcoming lines contain a character from 'i', 'd', 'f', 'e', or 't' followed by zero or one integer.
    - Character 'i' is followed by an integer (in the range $[-10^6, 10^6]$) separated by a space. In this operation, the integer is pushed to S, using the function PUSH().
    - Character 'd' is followed by an integer k (in the range $[1, 10^6]$) separated by a space. Perform the pop operation k times and print the popped elements in order, using the function POP().
    - Character 'f' is to check whether the Stack is full or not using the function STACK-FULL(S).
    - Character 'e' is to check whether the Stack is empty or not using the function STACK-EMPTY(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', if S is full, print `Full`.
    - For option 'd', print the deleted elements in order (separated by space). If S is empty, print `Empty`.
    - For option 'f', if S is full, then print `YES`, otherwise print `NO`
    - For option 'e', If S is empty, then print `YES`, otherwise print `NO`.

Sample Input:

```
10
i 8
i 10
d 1
i 12
d 4
e
i 18
e
t
```

Sample Output:

```
10
12 8 Empty Empty
YES
NO
```

2. Write a menu driven program to implement a STACK S using a singly linked list. Stack is to be declared as a structure with a pointer attributes *top* pointing to the top of the list. Each node in the list has an integer attribute *data* and a pointer attribute *next*. Keep *S.top* point to the first node (node at the front end) of the list. Your program must contain the following functions: (in function prototypes, S denotes a Stack and k denotes an integer value. All operations should run in $O(1)$ time.)

   - MAIN() - repeatedly reads a character 'i', 'd', 'e' or 't' from the terminal and calls the appropriate function until character 't' is entered.
   - CREATE-NODE(k) - creates a new node with *data* value k and returns a pointer to the created node. The attribute *next* of the new node should be set as `NULL`.
   - PUSH(S, k) - inserts a node with key k as the new top node of S.
   - POP(S) - returns the most recently inserted element from S and deletes it from S.
   - STACK-EMPTY(S) - checks whether the Stack is empty or not.

   Note:- For every PUSH() operation, a new node should be created by calling the CREATE-NODE() function.

   Input format:

   - Each line contains a character from 'i', 'd', 'e', or 't' followed by zero or one integer.
   - Character 'i' is followed by an integer k (in the range $[-10^6, 10^6]$) separated by space. In this operation, the node with k as data is inserted to the front end of S by calling the function PUSH() .
   - Character 'd' is followed by an integer k (in the range $[1, 10^6]$) separated by space. Perform the pop operation k times and print the popped elements in order, using the function POP().
   - Character 'e' is to check whether the Stack is empty or not by calling the function STACK-EMPTY(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 'd' print the deleted elements (separated by space). If S is empty, then print `Empty`.
   - For option 'e' if S is empty, then print `YES`. If S is not empty, then print `NO`.

Sample Input:

```
i 10
i 15
d 1
i 12
d 4
e
i 20
e
t
```

Sample Output:

```
15
12 10 Empty Empty
YES
NO
```

3. Write a menu driven program to implement a QUEUE Q using an array $A[0..n-1]$. The queue is to be declared as a structure with three attributes- *head*, *tail* and the 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 k denotes an integer. All operations should run in $O(1)$ time.)

   - MAIN() - repeatedly reads a character 'i', 'd', 'f' 'e' or 't' from terminal and calls the appropriate function until character 't' is entered.
   - QUEUEFULL(Q) - checks whether the Queue is full or not.
   - QUEUEEMPTY(Q) - checks whether the Queue is empty or not.
   - ENQUEUE(Q, k) - inserts the element k to the tail of Q.
   - DEQUEUE(Q) - deletes the element from the head of Q.

   Input format:

   - The first line of the input contains an integer $n \in [0, 10^5]$, the size of the array.
   - Upcoming lines contain a character from 'i', 'd', 'f' , 'e' or 't' followed by zero or one integer.
   - Character 'i' is followed by an integer (in the range $[-10^6, 10^6]$) separated by space. In this operation, this integer is inserted to the tail of Q.
   - Character 'd' is followed by an integer k (in the range $[1, 10^6]$) separated by space. The operation is to delete k elements from Q in the order of their insertion into Q and print the deleted elements.
   - Character 'f' is to check whether the Queue is full or not.
   - Character 'e' is to check whether the Queue is empty or not.
   - 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' if Q is full, then print `Full`.
   - For option 'd' print the deleted elements in order (separated by space). If Q is empty, then print `Empty`.
   - For option 'f', if Q is full, then print `YES`. If Q is not full, then print `NO`.
   - For option 'e', if Q is empty, then print `YES`. If Q is not empty, then print `NO`.

Sample Input:

```
5
i 20
i 38
d 3
e
i 40
e
i 35
i 42
f
i 33
i 27
f
t
```

Sample Output:

```
20 38 Empty
YES
NO
NO
Full
YES
```

4. Write a menu driven program to implement a QUEUE Q using a singly linked list. Queue is to be declared as a structure with two pointer attributes *head* and *tail*, pointing to the first node and the last node of the list respectively. Each node in the list is an object with an attribute *data* and a pointer attribute, *next*. Your program must contain the following functions: (In function prototypes, Q denotes a Queue and k denotes an integer (k need not be distinct). All operations should run in $O(1)$ time.)

   - MAIN() - repeatedly reads a character 'i', 'd', 'f' or 'e' from the terminal and calls the appropriate function until character 't' is entered.
   - CREATE-NODE(k) - creates a new node with data value k and returns a pointer to the new node. The *next* field should be set as NULL.
   - QUEUEEMPTY(Q) - checks whether the Queue is empty or not.
   - ENQUEUE(Q, k) - inserts a node with key k to the tail of Q.
   - DEQUEUE(Q) - deletes the first node from Q.

   Note:- For every ENQUEUE operation, a new node should be created by calling CREATE-NODE() function.

   Input format:

   - Each line contains a character from 'i', 'd', 'e' or 't' followed by zero or one integer.
   - Character 'i' is followed by an integer k (in the range $[-10^6, 10^6]$) separated by space. In this operation, the node with k as data is inserted to the tail of Q by calling the function ENQUEUE().
   - Character 'd' is followed by an integer k (in the range $[1, 10^6]$) separated by space. This operation is to delete the first k nodes from Q in the order of their insertion into Q and deleted nodes' keys is printed by calling the function DEQUEUE().
   - Character 'e' is to check whether the Queue is empty or not by calling the function QUEUEEMPTY().
   - 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 'd' print the deleted nodes' keys in order (separated by space). If Q is empty, then print `Empty`.
   - For option 'e', if Q is empty, then print `YES`. If Q is not empty, then print `NO`.

Sample Input:

```
i 30
i 48
d 3
e
i 50
e
t
```

Sample Output:

```
30 48 Empty
YES
NO
```