Algorithm Lab (Course Code: MC504)

Assignment - 4

Submission Deadline: Within class timing, (03/02/2023)
Total Marks: 50

Instructions:

- Proper indentation is mandatory.
- Program files must be compiled using linux gcc compiler.
- VERY IMPORTANT: You must add comments whenever necessary, to make the code understandable.
- Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
- Take inputs from users. Make necessary assumptions if required.
- ANSWER FILE: Source code: (file name) e.g. A4_Q1.c, A4_PP.c

---

Q1.

Write a C program to check if the input string is palindrome or not, using an array implementation of stack with push, pop, overflow (stack full), and underflow (stack empty) methods.

Q2.

While queues may be implemented in many ways, one of the simplest conceptually uses a singly- linked list with pointers head and tail giving access to the two ends:

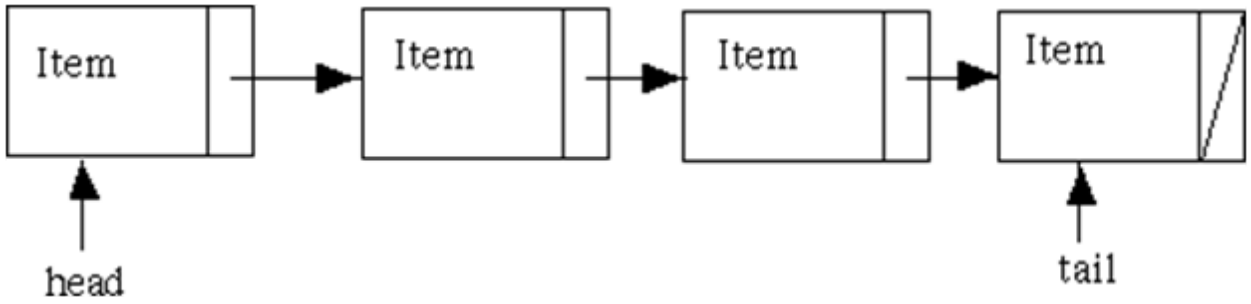

Fig: A Queue as a Singly-Linked List

With this perspective, an element on a queue is stored in a data field within a list node:

```
/* Maximum length of data */
#define strMax 20

typedef struct node{

 char data[strMax];
 struct node * next;

} queueNode;
```

The queue itself involves two pointer variables:

```
typedef struct {

  queueNode * head;

  queueNode * tail;

}stringQueue;

stringQueue queue;
```

With such a structure, each queue operation is reasonably straightforward:

void initializeQueue (stringQueue * queue) sets both head and tail fields to NULL.

int empty (stringQueue queue) tests head (or tail) against NULL.

int enqueue (stringQueue * queue, char* item) adds an element to the tail of the list: creating a new node, linking to it as the next item after the current last item, and updating the tail variable. In addition, if the queue had been empty, then head also must be updated to the new node. Recall that the enqueue returns the length of the string added to the queue.

char * dequeue (stringQueue * queue) retrieves the data in the head node, moves head to the second list item, deallocates the space of the old first node, and returns the data in that former head of the queue. In addition, if this item had been the last one on the queue, then tail must be updated to NULL.

Remember `enqueue` and `dequeue` require a little care to handle empty lists.

Tasks:

1. Implement the above kind of queue (and its operations) as described.
2. In the same file write a print function that prints all elements on a queue, from the head of the queue to its tail. (This function can be helpful in testing).
3. Test your program carefully. Think of a set of test cases that will thoroughly test your program.

Hint: Your `main()` function will initialize the queue. After that it will ask the user for four choices: `enqueue,dequeue,print,exit`. As long as the user is not choosing to exit, your program will give those four choices perpetually and do the desired operations. Your program must take care of the exceptional case when the user tries to dequeue from an empty list. Submit the source code file only.

Practice problem

Q3.

You are given an array A of Q integers and Q queries. In each query, you are given an integer i(1<=i<=n). Your task is to find the minimum index greater than i (1<=i<=n) such that:

1. Sum of digits of A_i is greater than the sum of digits of A_j.
2. $A_i <= A_j$

If there is no answer, then print **-1**.

Input format

1. The first line contains two numbers N and Q.
2. The next line contains N numbers.
3. Next Q lines contain Q queries.

Output format

Print the answer as described in the problem.

Constraint

(1<=N, Q <= 10^5)
(1 <= A_i <= 10^9)
(1 <= Q_i <= N)

Sample I/P

5 5
62 70 28 62 92

1
5
3
4
2

Sample O/P

2
-1
4
-1
-1

Explanation

In the first query 70 greater than 62 and sum of digits of 62 = 8 greater than sum of digits of 70=7 (8>7)