STACKS - QUEUES

A – Theory part

Important note: In this set of exercises, it is necessary to have the following operations

- *The **top()** operation reads the **element on the top of a stack***
- *The **front()** operation reads the **element at the front of a queue***
- *The **isEmpty()** operation checks the **emptiness of a queue or a stack***

A1. Match each of the following application with an appropriate choice from the list of available data structures, including stack, queue and linked list.

Applications	*Data structures*
Arithmetic expression evaluation	
When a resource is shared among multiple consumers.	
Managing nested function calls	
A grocery store decided that customers who come first will be served first	
Print jobs submitted by users of the system	
Reverse a string	
Manage a sorted list of integers	
A grocery list ordered by the occurrence of the items in the store	
Represent a polynomial by storing its coefficients and exponents	
Perform arithmetic operations on long integers	

A2. For each pseudo-code segments below, what output is displayed after it executes?

a.

```
Stack S;                          // A stack of integers
for (int i = 1; i < 10; i++)
     push(S, i);
while (!isEmpty(S)){
     cout << top(S) << endl;
     pop(S);
}
```

b.

```
Stack S;                          // A stack of integers
for (int i = 1; i < 10; i++)
     push(S, i);
while (!isEmpty(S))
     cout << top(S) << endl;      // run forever?
```

A3. Trace through the state of the stack S / queue Q in the following pseudo-code fragments.

a.

```
Stack S;                                    // A stack of strings
push(S, "happy");
push(S, "sad");
string st = top(S);
push(S, "numb");
push(S, st+"dle");
pop(S);
st = top(S);
pop(S);
push(S, st);
```

b.

```
Queue Q;                                    // A queue of integers
enqueue(Q, 5);
enqueue(Q, 7);
enqueue(Q, 13);
deQueue(Q);
int t = front(Q);
enqueue(Q, 12+t);
deQueue(Q);
enQueue(Q, front(Q));
deQueue(Q);
```

A4. Consider the following sequences of letters.

(i) E A S * Y * Q U E * * * S T * * * I O * N * * *

(ii) L A * S T I * N * F I R * S T * * O U * T * * * * * *

a. A letter means *push* and an asterisk means *pop* in the sequence, which is performed on an initially empty *LIFO stack*. Show the sequence of popped-out letters.

b. A letter means *enqueue* and an asterisk means *dequeue* in the following sequence, which is performed on an initially empty *FIFO queue*. Show the sequence of dequeued letters.

A5. Suppose that a client performs an intermixed sequence of enqueue and dequeue operations. The enqueue operations put the integers 0 through 9 in order onto the queue; the dequeue operations remove an element from the queue.

Following are sequences of integers that might involve in the removal. Which of those sequence(s) could not occur? Give your reasons.

a) 0 1 2 3 4 5 6 7 8 9

b) 4 6 8 7 5 3 2 9 0 1

c) 2 5 6 7 4 8 9 3 1 0

d) 4 3 2 1 0 5 6 7 8 9

A6. Suppose that a client performs an intermixed sequence of push and pop operations. The push operations put the integers 0 through 9 in order onto the stack; the pop operations remove an element from the stack.

The following are sequences of integers that might involve in the removal. Which of those sequence(s) could not occur? Give your reasons.

a) 4 3 2 1 0 9 8 7 6 5
b) 4 6 8 7 5 3 2 9 0 1
c) 2 5 6 7 4 8 9 3 1 0
d) 4 3 2 1 0 5 6 7 8 9
e) 1 2 3 4 5 6 9 8 7 0
f) 0 4 6 5 3 8 1 7 2 9
g) 1 4 7 9 8 6 5 3 0 2
h) 2 1 4 3 6 5 8 7 9 0

B – Coding part

B1. Write a program that reads in a sequence of characters and prints them in reverse order. The reversion should use a stack. E.g., reverse("duck") should return "kcud".

B2. Write a program that reads in a sequence of characters, and determines whether its parentheses, braces, and curly braces are "balanced." Hint: for left delimiters, push onto stack; for right delimiters, pop from stack and check whether popped element matches right delimiter.

B3. Write a program that reads in a positive integer and prints the binary representation of that integer. Hint: divide the integer by 2.

B4. The following data simulates the execution of a bank counter. Each line of data contains the arrival time and transaction processing time (in minutes) of a customer.

5 9
7 5
14 5
30 15
32 5
34 5

Note that at time 14 there is a tie between the execution of an arrival

Assume that this counter can only welcome one customer at a time and the waiting customer leaves with anger if he/she has to wait more than 10 minutes. Count the number of lost customers as incoming customer comes to a queue.