STACKS - QUEUES

In this material, please note the following functions as well as their functionalities.

- ***top(S)*** *and* ***front(Q)*** *read the element on the top of stack S and at the front of queue Q, respectively, without any modification on the content of these data structures.*
- ***isEmpty(D)****: check the emptiness of a queue or a stack*

Short Answer

1. * What does FIFO mean? What does LIFO mean?
2. * What is the difference between a static stack/queue and a dynamic stack/queue?
3. * Describe two common operations that all stacks perform.

   Similarly, describe two common operations for all queues.

4. ** Consider the below pseudo-code segment. What is output for the input `"geeksquiz"`?

```
declare a stack of characters
while ( there are more characters in the word to read ) {
      read a character
      push the character on the stack
}
while ( the stack is not empty ) {
      read the character on the top of the stack
      pop a character off the stack
      write the character to the screen
}
```

5. ** Consider the following pseudo-code functions. What does the function `func` do in each case?

-
```
void func(int n) {
    Queue Q;            // An empty queue of integers
    enqueue(Q, 0);
    enqueue(Q, 1);
    for (int i = 0; i < n; i++) {
        int a = front(Q);
        dequeue(Q);
        int b = front(Q);
        dequeue(Q);
        enqueue(Q, b);
        enqueue(Q, a + b);
        print(a);
    }
}
```

- ```
  void func(int n) {
      Stack S;                // An empty stack of integers
      while (n > 0) {
          push(S, n%2);       // Push the value of n%2 to stack S
          n = n/2;
      }
      // Run while Stack S is not empty
      while (!isEmpty(S)){    // Pop an element from S and print it
          print(top(S));
          pop(S);
      }
  }
  ```

- ```
  void func(Queue Q){           // Q: a queue of integers
      Stack S;                  // S: an empty stack of integers
      // Run while Q is not empty
      while (!isEmpty(Q)) {     // Push to S an item dequeued from Q
          push(S, front(Q));
          dequeue(Q);
      }
      // Run while Stack S is not empty
      while (!isEmpty(S)){      // Enqueue to Q an item popped from S
          enqueue(Q, top(S));
          pop(S);
      }
  }
  ```

6. ** What output is displayed after the each of following pseudo-code segment executes? Note that every segment is independent from the others.

   - ```
     Queue Q;                            // An empty queue of integers
     num1 = 4; num2 = 10; num3 = 2;      // Integer variables
     enqueue(Q, num2);
     enqueue(Q, num3);
     dequeue(Q);
     enqueue(Q, num1-num2);
     num1 = front(Q);
     dequeue(Q);
     num2 = front(Q);
     dequeue(Q);
     cout << num1 << " " << num2 << " " << num3 << endl;
     ```

   - ```
     Stack S;                            // An empty stack of integers
     for (int i = 1; i < 10; i++)
         push(S, i);
     while (!isEmpty(S))
         cout << top(S) << endl;
     ```

- ```
  Stack S;                              // An empty stack of integers
  for (int i = 1; i < 10; i++)
      push(S, i);
  while (!isEmpty(S)){
      cout << top(S) << endl;
      pop(S);
  }
  ```
- ```
  Stack S;            // A queue of integers
  int a = 22, b = 44;
  push(S, 2);
  push(S, a);
  push(S, a + b);
  b = top(S);
  pop(S);
  push(S, b);
  push(S, a - b);
  pop(S);
  while (!isEmpty(S)) {
      cout << top(S) << endl;
      pop(S);
  }
  ```

7. ** Trace through the state of the stack S (or queue Q) in the following pseudo-code fragments.

- ```
  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);
  ```
- ```
  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);
  ```

8. ** For each of the following applications, select an appropriate choice of available data structures, including stack, queue, and linked list. Explain why the selected choice is best while the others are less appropriate.

- Arithmetic expression evaluation
- When a resource is shared among multiple consumers.
- Managing nested function calls
- In a grocery store, 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

9. *** What output is displayed after the following pseudo-code program executes?

```
void print(Queue q){
    Queue qq = q;
    while (!isEmpty(qq)) {
        cout << front(qq) << endl;
        deQueue(qq);
    }
}
int main() {
    Queue Q;
    print(Q);
    enqueue(Q, "Bobo");   print(Q);
    enqueue(Q, "Billy");  print(Q);
    enqueue(Q, "Suzy");   print(Q);
    dequeue(Q);           print(Q);
    enqueue(Q, "Ari");    print(Q);
    dequeue(Q);           print(Q);
    return 0;
}
```

10. *** Is it possible to implement a stack using multiple queues? If yes, sketch such an idea. Similarly, how about implement a queue using multiple queues.

Fill-in-the-Blank

1. * Fill in each of the following blanks with an appropriate terminology.

- A stack is a data structure that stores and retrieves items in a _____ manner, while a _____ operates in a first-in-first -out manner .
- The two primary stack operations are _____ and _____, while those for a queue are _____ and _____.
- Remove a plate from the pile can be thought of an example of _____, while waiting in a line for a movie can be thought of an example of _____.

2. ** Fill in each of the following blanks with an appropriate value.

- If the sequence of operations – `push(1), push(2), pop, push(1), push(2), pop, pop, pop, push(2), pop` – are performed on a stack, the sequence of popped-out values is _____.
- If the sequence of operations – `push(a), pop, push(b), push(c), pop, push(d), pop, pop, push(e)` – are performed on a stack, the sequence of popped-out values is _____.
- The five items, `A, B, C, D,` and `E`, are pushed in a stack, one after other starting from `A`. The stack is popped four items and each element is inserted in a queue. The two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped-out item is _____.
- The seven elements `A, B, C, D, E, F` and `G` are pushed onto a stack in reverse order, i.e., starting from `G`. The stack is popped five times and each element is inserted into a queue. Two elements are deleted from the queue and pushed back onto the stack. Now, one element is popped from the stack. The popped-out item is _____.

3. ** A letter means push and an asterisk means pop in the following sequence, which is performed on an initially empty LIFO stack.

`E A S * Y * Q U E * * * S T * * * I O * N * * *`

The sequence of popped-out letters is _____.

Repeat the question for `L A * S T I * N * F I R * S T * * O U * T * * * * * *`

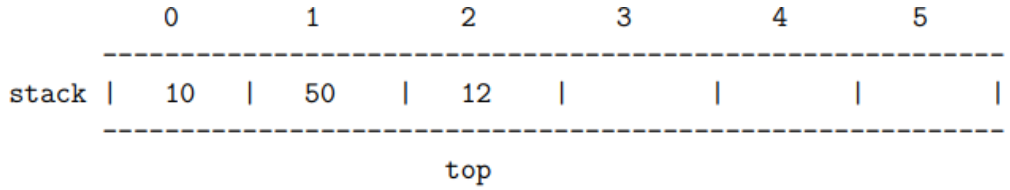
4. ** A letter means enqueue and an asterisk means dequeue in the following sequence, which is performed on an initially empty FIFO queue.

`E A S * Y * Q U E * * * S T * * * I O * N * * *`

The sequence of dequeued letters is _____.

Repeat the question for `L A * S T I * N * F I R * S T * * O U * T * * * * * *`

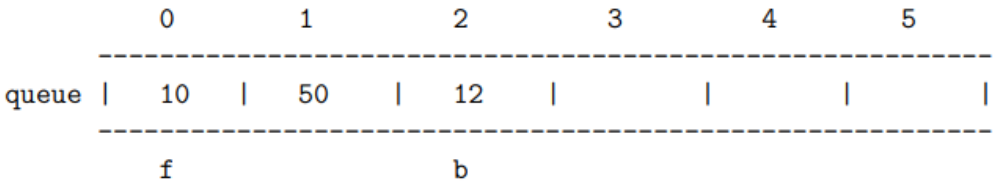
5. ** Given the following contents of an array implementation of a stack.



Show the content of the stack and the location of `top` after doing each of the following operations.

Operations	Location of `top`	Content of stack
`pop(stack);`		
`push(stack, 7);`		
`push(stack, 8);`		
`push(stack, 9);`		
`pop(stack);`		
`push(stack, 11);`		

6. ** Given the following contents of a circular array implementation of a queue.



Show the contents of the queue and the locations of `f` and `b` after doing each of the following operations.

Operations	Location of `f` and `b`	Content of queue
`dequeue(queue);`		
`enqueue(queue, 7);`		
`enqueue(queue, 8);`		
`enqueue(queue, 9);`		
`dequeue(queue);`		
`enqueue(queue, 11);`		

7. ** Stack `A` has the entries `a`, `b`, and `c` (with `a` on top). Stack `B` is empty. An entry popped out of stack `A` can be printed immediately or pushed to stack `B`. An entry popped out of the stack `B` can be only be printed. Identify whether each of the below permutations of `a`, `b`, and `c` is possible.

Permutations	Possible/Impossible?	Explanation
`b a c`	_____	_____
`b c a`	_____	_____
`c a b`	_____	_____
`a b c`	_____	_____

8. ** Which of the following permutation can be obtained in the same order using a stack assuming that input is the sequence `5, 6, 7, 8, 9` in that order? Justify your answer.

Permutations	Possible/Impossible?	Explanation
`7, 8, 9, 5, 6`	_____	_____
`5, 9, 6, 7, 8`	_____	_____
`7, 8, 9, 6, 5`	_____	_____
`9, 8, 7, 5, 6`	_____	_____

9. *** 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? Justify your answer.

Permutations	Possible/Impossible?	Explanation
`0 1 2 3 4 5 6 7 8 9`	_____	_____
`4 6 8 7 5 3 2 9 0 1`	_____	_____
`2 5 6 7 4 8 9 3 1 0`	_____	_____
`4 3 2 1 0 5 6 7 8 9`	_____	_____

10. *** 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. Following are sequences of integers that might involve in the removal. Which of those sequence(s) could not occur? Justify your answer.

Permutations	Possible/Impossible?	Explanation
`4 3 2 1 0 9 8 7 6 5`	_____	_____
`4 6 8 7 5 3 2 9 0 1`	_____	_____
`2 5 6 7 4 8 9 3 1 0`	_____	_____
`4 3 2 1 0 5 6 7 8 9`	_____	_____
`1 2 3 4 5 6 9 8 7 0`	_____	_____
`0 4 6 5 3 8 1 7 2 9`	_____	_____
`1 4 7 9 8 6 5 3 0 2`	_____	_____
`2 1 4 3 6 5 8 7 9 0`	_____	_____

True or False

Choose T (True) or F (False) for each of the following statements and then briefly explain in one or two sentences.

1. T F The push operation inserts an element at the end of a stack.
2. T F The enqueue operation remove an element from the rear of a queue.
3. T F A stack is a data structure that stores and retrieves elements in a FIFO manner.
4. T F A queue is a data structure in which all insertions and deletions are made respectively at rear and front.
5. T F In dynamic queue, the enqueue operation is equivalent to adding a new node to the rear of the linked list.
6. T F In dynamic stack, an uninitialized top pointer signifies that the stack is empty.
7. T F There is no need to specify the starting size of a dynamic stack/queue.
8. T F The two stack operations, push and pop, are semantically equivalent to the two queue operations, enqueue and dequeue, respectively.
9. T F An empty queue can be signified by setting both front and rear indices to 0.
10. T F The queue implementation as a circular array provides dynamic queues.

Algorithm Workbench

1. * Prepare an implementation of stack using array. Repeat the question for a queue.
2. * Prepare an implementation of stack using singly linked list. Repeat the question for a queue.
3. ** Implement the pseudo-code given in the Fill-in-the-Blank section, Question 3.
   Repeat the question for the pseudo-code given in the Fill-in-the-Blank section, Question 4.
4. ** Write code to receive a sequence of characters and print them in reverse order. The reversion should use a stack. For example, reverse("duck") should return "kcud".
5. ** Write a program that reads in a positive integer and prints the binary representation of that integer. For example, the integer 10 has its binary representation of 1010.
6. ** Write code to input a sequence of characters and determine 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.*
7. ** Consider an infix (arithmetic) expression, whose operands are integers and operations are in the set of {+, –, *, /}. Write code to first convert the expression from infix to postfix and then evaluate the expression. For example, the infix expression, 3 + 4, corresponds to the postfix expression, 3 4 +, and it is evaluated as 7.
8. ** The following data simulates the execution of a bank counter. Each line of data contains the arrival time and the 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.

9. *** You are tasked to write a queue simulation program for a new supermarket. It is known that in this supermarket, they employ a VIP system which allows a VIP customer to cut into the first position of the current serving queue at the cash register. If there are more than 1 VIP customer appearing at the queue, the VIP customers will be arranged in such a way that the VIP customer who joined the queue earlier will be at the front of the VIP customer who came later.
   Below is the input format which the system uses for the simulation.

```
X (a non-negative integer, the number of customers initially in the queue)
Name_of_customer-1
Name_of_customer-2
…
Name_of_customer-X
Y (a positive integer, the number of new customers joining the queue)
Customer_type Name_of_new_customer-1
Customer_type Name_of_new_customer-2
…
Customer_type Name_of_new_customer-Y
Search_name
```

Initially the system reads X, a non-negative integer indicating the number of customers initially in the queue. There can be no customer in the queue initially. The system then reads X number of names and inserts the names into a FIFO queue. After that, the system reads Y, a positive integer indicting the number of new customers joining the queue. This is followed by Y lines, each line representing a new customer with the following information: customer-type (which signifies whether the customer is a VIP customer or a regular customer) and name. All names are unique. Finally, a search name is read. The system then reports how many customers are ahead of this search name in the queue.

You may assume the following.

- All names are single words, e.g., "James", "Cindy" and they are case-sensitive.
- Customer-type is either "VIP" or "Regular". All customers in the initial queue are assumed to be of "Regular" type.
- There can be no customer in the initial queue.
- The search name will always be present in the queue.

For example,

Input	Output
2 James Cindy 4 Regular Chan VIP Kong Regular Lee VIP May **Lee**	Final queue: Kong, May, James, Cindy, Chan, Lee There are 5 customers ahead of Lee

10. *** Assume that you are not as careful programmer as you are and you frequently forget to close your brackets while you program – (, {, [-. Also, because you have not paid attention in writing to write your code in a proper fashion you find it difficult to find where the mistake is. To help you while programming, construct a program that takes as input a programming file. The program should check if all the brackets opened are correctly closed. For this to happen, there should be as many opening brackets as closing ones. Also, the bracket type closing must be the same as the one LAST opened. Amend your program with the following functionality: when an error is identified write on the console the line and column that the error was found and specify the expected type of closing bracket.