ASSIGNMENT-1

CSE-204, COURSE: DATA STRUCTURE SESSIONAL

Initialize a dynamic array and ask user to give input the initial size and elements of the array. Also create the following menu:

1. Insert element at the first position of the array
2. Insert element at the last position of the array
3. Insert element at any position of the array
4. Delete an element from the first position of the array
5. Delete an element from the first position of the array
6. Delete an element from any position of the array
7. Update any value of any position of the array
8. Quit

Create different functions for the different options. Make sure, the size of the array is updated after each insert and delete.

```
Enter the size of the array: 5
Enter the elements of the array: 10 12 5 70 11
The resultant array: 10 12 5 70 11
Choose one of the option:
    1. Insert element at the first position of the array
    2. Insert element at the last position of the array
    3. Insert element at any middle position of the array
    4. Delete an element from the first position of the array
    5. Delete an element from the first position of the array
    6. Delete an element from any position of the array
    7. Update any value of any position of the array
    8. Quit
Enter option: 1
Enter value: 4
The resultant array: 4 10 12 5 70 11
Enter option: 2
Enter value: 8
The resultant array: 4 10 12 5 70 11 8
Enter option: 3
Enter position: 2
Enter value: 50
The resultant array: 4 50 10 12 5 70 11 8
Enter option: 4
The resultant array: 50 10 12 5 70 11 8
Enter option: 5
The resultant array: 50 10 12 5 70 11
Enter option:6
Enter position: 8
Entered position is not a valid one.
Enter option:6
Enter position: 4
The resultant array: 50 10 12 70 11
Enter option:7
Enter position: 4
Enter value: 9
The resultant array: 50 10 12 9 11
Enter option:8
```

LAB-02
Data Structure
CSE-204
Assignment on Binary Search
Military Institute of Science and Technology

1) Implement the following function:

void frequency(int n)

The function will take an integer *n* and find how many times it has occurred in that array. For example, if the arraylist contains 1,2,3,3,3,3,4,5 then after calling frequency(3) the output will be 4.

2) Implement the following function:

void insert_after_val (int val, int num_of_inputs)

The function will take an integer value val which has to be searched by binary search and num_of_inputs will say how many integers will be added by user in the arraylist after the value val. Insertion must be maintain a sorted arraylist. For example, if the arraylist contains 1, 2, 3, 4, 5, 7 then after calling insert_after_val (3,2) the output will be 1,2,3,x1,x2,5,7.

NB: While taking an input to insert in the array list, it will be inserted when it must be greater than its previous value and less than its next value. Otherwise it will show an error massage.
For example, if the arraylist contains 1, 2, 3, 4, 5, 7 then after calling insert_after_val (3, 1).
Insert: 8

Output: value can't be inserted.

LAB-02
Data Structure
CSE-204
Assignment on Binary Search
Military Institute of Science and Technology

Mamun went to fight for Coding Club. There were N soldiers with various powers. With power P, Mamun can kill all the soldiers whose power is less than or equal to P(<=P). Such that there will be N soldiers to fight. As Mamun is weak in mathematics, help him to count the number of soldiers that he can kill and total sum of their powers.

For example:

Number of the soldier: N= 7
Soldier Power successively: 1 5 7 9 9 11 15
~~Bishu~~'s power: 7
MAMUN
Output:
Number of soldier killed: 3
Total power of the killed soldier: 13

Evaluation:

1. Suppose you are working at Microsoft. Bill Gates has assigned you in the Microsoft Word development team. Your first task is to make a plugin which will act as a validation checker for mathematical expressions.

   The behavior of your program will be like this:

   a. **[{ ( ) }]** : Valid Expression
   b. **[{] }** : Invalid Expression.
   c. **[{** : Invalid Expression.

Lab Task:

2. Queue is a linear data structure which follows a particular order in which the operations are performed. The order may be FIFO (First In First Out) or LILO (Last In Last Out).

   Mainly the following four basic operations are performed in the stack:

   d. **Enqueue**: Adds an item in the queue. If the queue is full, then it is said to be an Overflow condition.
   e. **Dequeue**: Removes an item from the Queue. The oldest item in the queue will be dequeued. If the queue is empty, then it is said to be an Underflow condition.
   f. **isEmpty**: Returns true if queue is empty, else false.
   g. **size**: Returns the length.

   Implement a queue of Integer values. You can define the size of a queue in the code (#define MAX 5). (NB: Implement it using object oriented approach)

Home Assignment:

1. Implement an cost efficient/ optimized version of queue.

LAB-02
Data Structure
CSE-204
Military Institute of Science and Technology

1) Implement the following function:

void LeftRotate(int n)

The function will take an integer *n* and left rotate the array *n* times. For example, if the arraylist contains 1,2,3,4,5 then after calling LeftRotate(3) the list will look like 4,5,1,2,3

2) Implement the following function:

void Reverse(int start, int end)

The function will take two integer *start*, *end* and reverse the array *from start* to *end*. For example, if the arraylist contains 1,2,3,4,5,6 then after calling Reverse(2,4) the list will look like 1,4,3,2,5,6.

3) Implement the following function:

void Max_Min (int start, int end)

The following function will find the maximum and minimum number in the arralist. Fpr example, if the arraylist contains 1,9,20,4,8,0 then after calling the function Max_Min(0,5) it will give output :

Max: 20
Min: 0

Military Institute of Science and Technology (MIST)
Department of Computer Science and Engineering
CSE-204, Data Structure and Algorithm-I, Evaluation-1, Week-2, Date: 11 Feb, 2020

Create a dynamic array in C language of integer type and write necessary C codes to perform the following operations on the dynamic array.

1. InsertAt
2. DeleteFirst
3. PrintArray
4. Quit

Users will be asked to enter the initial capacity of the dynamic array for initialization. Then the above menu will be prompted for performing any of the operations as described below.

Menu	Description	Parameter	Return
InsertAt	A new element will be inserted in a specific position of the dynamic array.	Element, Position	True for success and false for failure.
DeleteFirst	First element of the dynamic array will be deleted.	No parameter	True for success and false for failure.
PrintArray	Print all the elements of the dynamic array.	No parameter	void
Quit	Terminate the program.	No parameter	exit

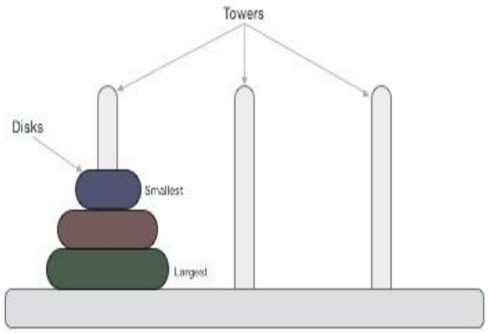
Sample Output: Green color means input given by user and blue color gives output.

Enter the capacity of the array: **5** Menu: 1. InsertAt 2. DeleteFirst 3. PrintArray 4. Quit Enter your choice: **1** Enter element and position: **4 0** Success Menu: 1. InsertAt 2. DeleteFirst 3. PrintArray 4. Quit Enter your choice: **1** Enter element and position: **51 1** Success	Menu: 1. InsertAt 2. DeleteFirst 3. PrintArray 4. Quit Enter your choice: **3** Array Elements: **4 51** Menu: 1. InsertAt 2. DeleteFirst 3. PrintArray 4. Quit Enter your choice: **2** Success	Menu: 1. InsertAt 2. DeleteFirst 3. PrintArray 4. Quit Enter your choice: **3** Array Elements: **51**

Military Institute of Science and Technology (MIST)
Department of Computer Science and Engineering
CSE-204, Data Structure and Algorithm-I, Evaluation-3, Week-6, Date: 10 March, 2020

Problem-1: Tower of Hanoi **30 Marks**

Tower of Hanoi, is a mathematical puzzle which consists of three towers (pegs) and more than one rings is depicted in figure.
These rings are of different sizes and stacked upon in an ascending order, i.e. the smaller one sits over the larger one. There are other variations of the puzzle where the number of disks increase, but the tower count remains the same.



The mission is to move all the disks to some other tower without violating the sequence of arrangement. A few rules to be followed for Tower of Hanoi are –

- Only one disk can be moved among the towers at any given time.
- Only the "top" disk can be removed.
- No large disk can sit over a small disk.

The pseudocode of iterative approach of solving the problem is given here:

```
move(Stack s1, Stack s2){
      move top of s1 to s2 or top of s2 to s1 if legal move possible
}
int main() {
       Stack src, dest, aux;
       n := number of disks; /// take input from user
       if n % 2 == 0
              swap(dest, aux) /// swap the address of stack locations
       for(1 to 2^n − 1) {
              if i % 3 == 1
                     move(src, dest);
              if i % 3 == 2
                     move(src, aux);
              else
                     move(dest, aux);

       }
}
```

*****Follow the above pseudocode and solve this problem using the concept of stack.**

Sample Input	Sample Output
Enter no. of Disks: 3	<1,2,3> <> <> <2,3> <1> <> <3> <1> <2> . . . . . . <> <> <1,2,3>

Tentative Marks Distribution

S/N	Tasks	Marks
1.	Stack Implementation	10
2.	Move function Implementation	7
3.	Pseudocode Implementation	5
4.	swap () function	5
5.	Output as shown	3
	Total	30

LAB-06
Data Structure
CSE-204
Evaluation on Link List
Military Institute of Science and Technology

1. Create a link list that can store any number of values. And implement the following operation:
   a. Display the list like following:

Value of the Node	Address of the Node	Address of the Next Node
10	0*1111	0*2222
20	0*2222	0*3333

And so on…..

   b. Find the average of the list.
   c. Search a particular item from the list and if found count the number of occurrences and show their address.

2. **Josephus Problem:**

Suppose there are n people numbered 1 to n standing around a circle. Every second remaining person is eliminated until only one survives. Given the number of persons in the initial setup, the problem is to find the number of survivor, J(n). For example, for n=10, the elimination order is 2, 4, 6, 8, 10, 3, 7, 1, 9, so 5 survives. So J(10)=5.

Hints:

A. Create a circular link list that contains values from 1 to n. Take n as a user input.
B. Start traversing from head node and eliminate every 2nd node until only one node remains.
C. Also show the elimination order.

LAB-05

CSE-204

Military Institute of Science and Technology

Evaluation:

Suppose you are a lazy developer (which you really are). After a tiresome effort, somehow you have managed to implement a stack class (with push, pop and isEmpty method) successfully. Now, your boss wants you to develop a queue class. But you don't want to implement it from the scratch. Rather you prefer that you will reuse the code of stack class.

Stack's push and Queue's enqueue works in the same way (as both of them are inserting new element just after the last element present in the array). So your push method will work as enqueue. But the policy of pop and dequeue is quite opposite. You have decided that, you won't write dequeue method in your stack class. Rather you would do it manually in the main function (remember, you do not know the size of your stack).
(Hints: is it possible to implement a queue with the help of two stacks?)

Lab Task:

1. Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.
   Mainly link list should have the following functions:

   1. **Create list**
   2. **Display list**
   3. **Insert data in list:**
      a. Insert in the first position
      b. Insert in the last position
      c. Insert in a particular position
   4. **Delete data from list:**
      a. Delete from the first position
      b. Delete from the last position
      c. Delete from a particular position

Home Assignment:

1. *Implement all the functions of a link list.*
2. *Implement a circular link list.*

Lab Task:

1. Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

   Mainly the following four basic operations are performed in the stack:

   a. **Push**: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
   b. **Pop**: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
   c. **Peek or Top**: Returns top element of stack.
   d. **isEmpty**: Returns true if stack is empty, else false.

   Implement a stack of Integer values. You can define the size of a stack in the code (#define MAX 10;). (NB: Implement it using object oriented approach)

Home Assignment:

1. Implement a method named **merge (**Stack s**)** which will take a stack as an input and merge it with the corresponding stack. You must maintain the overflow condition (merge until the corresponding stack is not overflowed).
   Example: Let us say, your stack1 is containing 1 2 3 4
   Your stack2 is containing 5 6 7 8
   After merging stack2 to stack1, the state of stack1 will be 1 2 3 4 8 7 6 5