Previous Lab Discussion

Bubble Sorting

Bubble sort is a simple sorting algorithm that works by repeatedly comparing adjacent elements and swapping them if they are in the wrong order. The algorithm starts at the beginning of the array and compares the first two elements. If the first element is greater than the second element, the two elements are swapped. The algorithm then moves on to the next two elements and repeats the process. This process continues until the end of the array is reached. The algorithm then repeats the entire process again, starting at the beginning of the array. This continues until no more swaps are needed, indicating that the array is sorted.

Selection Sorting

Selection sort is another simple sorting algorithm that works by finding the smallest element in the unsorted array and swapping it with the first element in the array. The algorithm then repeats this process for the remaining elements in the array, finding the smallest element in the unsorted subarray and swapping it with the next element in the sorted array. This process continues until the entire array is sorted.

Insertion Sorting

Insertion sort is a sorting algorithm that works by inserting each element of the unsorted array into its correct position in the sorted array. The algorithm starts at the second element in the unsorted array and compares it to the first element in the sorted array. If the second element is smaller than the first element, the second element is inserted before the first element. The algorithm then moves on to the next element in the unsorted array and repeats the process. This process continues until the end of the unsorted array is reached.

Current Lab Discussion

Stack

A stack is a linear data structure that follows the Last In First Out (LIFO)principle. This means that the last element added to the stack is the first element to be removed. Stacks are typically implemented using arrays or linked lists.

The basic operations on a stack are:

- **Push:** Adds an element to the top of the stack.
- **Pop:** Removes the top element from the stack.
- **Display:** Displays all the elements in the stack, in order from top to bottom.

TASK-04

Create a menu driven program for Stack.

Use class based implementation for Stacks. Take inputs from user.

Code:

- **.h File**

```
#include <iostream>
using namespace std;

class IntStack{

  private:
    int *stackArray;
    int stackSize;
    int top;
    int num;

  public:
```

```
IntStack(int size){
  stackArray = new int[size];
  stackSize = size;
  top = -1;
}

~IntStack(){
  delete [] stackArray;
}

void push(){

  if(isFull()){
    cout << "\nThe stack is full.\n";
  }
  else{
    top++;

    cout << "\nEnter the Number you want to PUSH: ";
    cin >> num;
    stackArray[top] = num;
    cout << "\nNumber is Pushed into Stack!!\n";
  }
}




void pop(){

  if(isEmpty()){
    cout << "\nStack Overflow!!\n";
  }
  else{
    cout << "\nThe popped element is: " << stackArray[top] << endl;
```

```
    top--;
  }
}


bool isFull(){

  if(top == stackSize - 1){
    return true;
  }
  else{
    return false;
  }
}

bool isEmpty(){

  if(top == -1){
    return true;
  }
  else{
    return false;
  }
}


void display(){

  if(isEmpty()){
    cout << "\nStack Underflow!!\n";
  }
  else{
    cout << "\nThe stack elements are: ";
```

```
    for(int i = top; i >= 0; i--){  // for(int i = 0; i <= top; i++)
      cout << stackArray[i] << " ";
    }
    cout << endl;
  }

 }

};
```

- **.cpp file**

```
#include "stack.h"
#include <iostream>
using namespace std;

int main()
{
 int size, num, choice;
 cout << "Enter the Size of Stack: ";
 cin >> size;
 IntStack ist(size);

 do {
  cout << "\n\nPress 1 to PUSH number into the stack.\n";
  cout << "Press 2 to POP number from the stack.\n";
  cout << "Press 3 to DISPLAY the stack.\n";
  cout << "Press 4 to Exit.\n";
  cout << "\n\nEnter Your Choice: ";
  cin >> choice;

  if (choice == 1)
      {
   cout << "Enter the total numbers you want to push: ";
```

```
    cin >> num;
    for (int i = 0; i < num; i++)
        {
     ist.push();
    }
  }
      else if (choice == 2)
      {
    ist.pop();
  }
      else if (choice == 3)
      {
    ist.display();
  }
      else if (choice == 4)
      {
    break;
  }
      else
      {
    cout << "Invalid Choice...!!";
  }

 }
 while (true);
}
```