Lab 1: Understanding the Stack

a. Explain Data structure, basic operations of data structures, ADT and it's necessity.

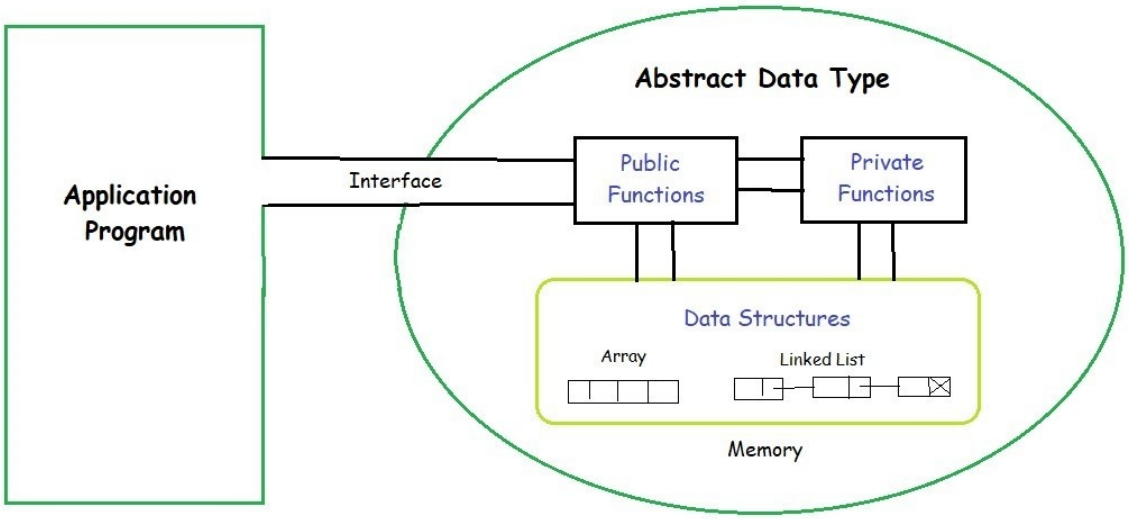
Data structure is a collection of data organization, management, and storage format that enables efficient access and modification. More precisely, a data structure is a collection of data values, the relationship among them, and the functions or operations that can be applied to the data. Hence, It is a way of storing and organizing data efficiently in a computer such that the required operations on them can be performed efficiently with respect to time as well as memory.

Basic operations:

1. Insertion: It means addition of a new data element in a data structure.
2. Deletion: It means removal of a data element from a data structure if it is found.
3. Searching: It involves searching for the specified data element in a data structure.
4. Traversal: Traversal of a data structure means processing all the data elements present in it.
5. Sorting: Arranging data elements of a data structure in a specified order is called sorting.
6. Merging: Combining elements of two similar data structures to form a new data structure of the same type, is called merging.

Abstract Data Type (ADT): It is defined as a type (or class) of objects whose logical behavior is defined by a set of values and a set of operations. It is called "abstract" because it gives an implementation independent view. The process of providing only the essentials and hiding the details is known as abstraction.

Abstract data types (ADTs) are used in large-scale programming. It separates the users from underlying structure of data structure and operations on them by providing users with interactive interface such as providing insertion and lookup operations.



b. Explain the algorithm for push and pop operation in stack.

Push operation: The process of adding a new data element onto stack is known as push operation.

Algorithm:

Step 1: Start
Step 2: Check if the stack is full
Step 3: If stack is full, state overflow condition and exit
Else increments "top" to point next empty space
Step 4: Add data element to the stack location, where top is pointing
Step 5: Stop

Pop operation: The process of accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead top is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element and frees memory space.

Algorithm:

Step 1: Start
Step 2: check if the stack is empty
Step 3: If the stack is empty, state underflow condition and exit

Else access the data at which “top” is pointing
Step 4: Decreases the value of top by 1
Step 5: Stop

Implementation of operation of Stack using C++

```
#include <iostream>
#include <queue>
#define SIZE 5

using namespace std;

class Stack
{
	int stack[SIZE], top;
public:

	Stack(){
		top = -1;
	}	//Constructor for initializing top

	int push(int num){
		if(top == SIZE-1){
			cout << "STACK OVERFLOW" << endl;
		}	//Stack is full
		else{
			top++;
			stack[top] = num;
			cout << num << " is push" << endl;
		}	//incrementing top by 1
		return 0; //I want to know this annoying line of code why exist??
				//I don't want to return anything then simply why can't we ignore it?
				//why does compiler give warning?
	}
	int pop(){
		int num;
		if(top == -1){
			cout << "STACK UNDERFLOW" << endl;
		}	//Stack is empty
		else{
			num = stack[top];
			top--;
			cout << "ELEMENT DELETED: " << num << endl;
		}	//decreasing top by 1
		return 0;
	}
	int traverse(){
		if(top == -1){
```

```
            cout << "STACK IS EMPTY" << endl;
        }
        else{
            cout << "STACK ELEMENTS";
            for(int i = top; i >= 0; i--){
                cout << stack[i] << endl;
            }
        }
        return 0;
    }
    bool Search(int num){
        int i = top;
        if(top == -1) cout << "there is no element in the stack" << endl;
        for(i; i >= 0; i--){
            if(stack[i] == num){
                return true;
                break;
            }   //return true if the value is present in stack
        }
        return false;   //return false if the value is not present in stack
    }
};

int main()
{
    Stack obj1;
    int ch, val;
    bool test;
    cout << "1) Push in stack" << endl;
    cout << "2) Pop from stack" << endl;
    cout << "3) Display stack" << endl;
    cout << "4) Search" << endl;
    cout << "5) Exit" << endl;
    do{
        cout << "Enter choice:";
        cin >> ch;
        if(ch == 1){
            cout<<"Enter value to be pushed:" << endl;
            cin>>val;
            obj1.push(val);
        }
        else if(ch == 2)obj1.pop();
        else if(ch == 3)obj1.traverse();
        else if(ch == 4){
            cout << "Enter value to be searched:" << endl;
            cin >> val;
            test = obj1.Search(val);
            if(test == true) cout << val << " is present" << endl;
            else cout << val << " is not present" << endl;
        }
```

```
            else cout << "Please enter the valid number" << endl;

        }while(ch != 5);
        return 0;
}
```

File name: stack.cpp