**STACK Assignment**

• **Stack Data Structure (Using Array)**  
In this program, the stack is implemented using an array. A stack is a linear data structure that follows the **LIFO** (Last In, First Out) principle. This means the element that is inserted last will be removed first.

• **Dynamic Memory Allocation**  
The array for the stack is created dynamically using the new keyword. This allows the user to define the stack size at runtime.



• **Push Operation**  
When a new element is inserted into the stack, the program first checks whether the stack is full or not. If the stack is full, it prints a "Stack Overflow" message. Otherwise, it increments the top and stores the new value at that position.



• **Pop Operation**  
The pop operation removes the element from the top of the stack. If the stack is empty (i.e., top == -1), it prints a "Stack Underflow" message. Otherwise, it simply decreases the top to remove the element.



• **Peek Operation**  
The peek function is used to view the top element of the stack without removing it. If the stack is empty, it shows an appropriate message. Otherwise, it returns the top value.



• **isEmpty Function**  
This function checks whether the stack is empty or not. If top == -1, it means the stack has no elements.

• **Size Function (IsSize)**  
This function returns the current number of elements in the stack. It simply returns top + 1.

• **Display Function**  
The display function prints all elements of the stack from top to bottom (in LIFO order). If the stack is empty, it prints a message saying so.

• **Search Function**  
The search function is used to find a specific value in the stack. It performs a simple linear search from index 0 to top. If the value is found, it prints the index; otherwise, it shows a "not found" message.



• **Reverse Function**  
The reverse function does not actually reverse the stack but displays the elements in reverse order (i.e., similar to FIFO). It runs a loop from index 0 to top and prints the elements.



• **Object-Oriented Programming (OOP) Concepts**  
This code makes use of OOP concepts such as classes, constructors, encapsulation, and data hiding. All stack operations are written inside the class as member functions.



• **Modular Approach**  
Each operation (such as push, pop, peek, etc.) is written in a separate function, which makes the code more modular and easier to understand.

============================================================

**Real-Life Examples of Stack:**

**=>Stack of Plates in a Canteen**

Imagine you have a stack of plates in a canteen.

* You **add** plates one by one on top.
* When someone takes a plate, they always take the **top one first**.
* You can’t take the bottom plate without removing the ones above.

This is exactly how a stack works — **Last In = First Out**

**CODE:-**

#include<iostream>

using namespace std;

class Stack{ //stack ki class bna li

int \*arr; // ye pointer h jo stack k array ko point krega

int size; //for the size of stack

int top; //top element of the stack

public:

Stack(int s){ //s ak prameter se size define hogi

size=s; //size set k diye user se leke

top=-1; //top element -1 h iska mtlab stack khali h

arr=new int[s]; //dynamic arrey create kiya give size ka or element store kra diye

}

void push(int val){ //ak function bnaya push ka jo value dalne ka kam krega

if(top==size-1){ //aager stack full h toh overflow ki condition ho jayegi or element nhi dal sakte

cout<<"stack overflow"<<endl;

return; //overflow hote hi return kr jayegaa

}

top++; //value dalne ke liye top ko bdayege

arr[top]=val; //top ko nyii value dedege

cout<<"pused "<<arr[top]<<" into the stack"<<endl; //conformation

}

void pop(){ //ak function bnaya pop ka jo value htane ka kam krega

if(top==-1){ // aager stack me kuch hoga hi nhi toh kise htge underflow ka case

cout<<"stack underflow"<<endl;

return;

}

cout<<"poped "<<arr[top]<<" from the stack"<<endl; //jo top element tha usko dikhayege ki ye pop kra h

top--; //top ko piche le jayege phle wala top aapne aap hat jayega

}

int peek(){ //ak function bnaya peek ka jo top ki value btayega

if(top==-1){ //aager stack khali huii toh top ki koi value hi nhi hogi

cout<<"stack is empty"<<endl;

return -1; //return kra diya -1 (ase hi mere mn me aagai hehehe..)

}

return arr[top]; //ye return krayega top ki value

}

bool isempty(){ //ak function bnaya Isempty ka jo stack khali hone pr true(1) dega else (0)

return top==-1; //top==-1 ka mtalb h stack is empty

}

int IsSize(){ //ak function bnaya IsSize ka jo size of stack btayega

return top+1;

}

void Display(){ //ak function bnaya Display ka jo stack ke element ko display krega

if(isempty()){ //aager stack khali ho toh empty display hoga

cout<<"stack is empty"<<endl;

return;

}

for(int i=top;i>=0;i--){ //ak loop chalayege jo top se piche ki orr chalega

cout<<arr[i]<<" "; //ak ak element print ho jayega jab tak index 0 na ho

}

cout<<endl;

}

void Search(int key){ //ak function bnaya search ka jo stack me se value dundega

// for(int i=top;i>=0;i--){

// if(arr[i]==key){

// cout<<i<<endl;

// }

// }

for(int i=0;i<=top;i++){ //ak loop chalayega jo index ko badayega jab tak user key k equal na ho jaye

if(arr[i]==key){ //aager key equal hoti h koi value toh index print kr dege

cout<<"Index "<<i<<" found value "<<arr[i]<<endl;

return;

}

}

cout<<"key not found in stack"<<endl; //aager key nhi mile toh not found print kra dege

}

void Reverse(){ //ak function bnaya Reverse ka jo stake ke elements ko reverse krege

if(top==-1){ //ager stack khali h toh kuch nhi reverse nhi kr payege

cout<<"stack is empty,no need to reverse";

return;

}

for(int i=0;i<=top;i++){ //loop chala dege index 0 se top tak or values print kra dege

cout<<arr[i]<<" ";

}

cout<<endl;

}

};

int main(){

Stack a(5); //a ak object liya h class ka or size of stack define kr di!!

a.Display();

a.push(1); //pused 1 into the stack

a.push(2); //pused 2 into the stack

a.push(3); //pused 3 into the stack

a.push(4); //pused 4 into the stack

a.pop(); //poped 4 from the stack

cout<<a.IsSize()<<endl; //3

a.Search(2); //Index 1 found value 2

a.push(4); //pused 4 into the stack

a.Display(); //4 3 2 1

a.Reverse(); //1 2 3 4

cout<<a.peek()<<endl; //4

cout<<a.isempty()<<endl; //0 (false)

cout<<a.IsSize()<<endl; //4

}