**Implementation of Stack**

**RA1911003010696**

**Sai Ram Pendyala**

**Aim:**

To implement stack in c++.

**Algorithm:**

1. push: Inserts a new element at the top of the stack, above its current top element.
2. pop: Removes the top element on the stack, thereby decrementing its size by one.
3. isEmpty: Returns true if the stack is empty, i.e., its size is zero; otherwise, it returns false.
4. isFull: Returns true if the stack is full, i.e., its size has reached maximum allocated capacity; otherwise, it returns false.
5. peek: Returns the top element present in the stack without modifying the stack.
6. size: Returns the count of elements present in the stack.

**Program:**

#include <bits/stdc++.h>

using namespace std;

#define MAX 1000

class Stack {

int top;

public:

int a[MAX]; // Maximum size of Stack

Stack() { top = -1; }

bool push(int x);

int pop();

int peek();

bool isEmpty();

};

bool Stack::push(int x)

{

if (top >= (MAX - 1)) {

cout << "Stack Overflow";

return false;

}

else {

a[++top] = x;

cout << x << " pushed into stack\n";

return true;

}

}

int Stack::pop()

{

if (top < 0) {

cout << "Stack Underflow";

return 0;

}

else {

int x = a[top--];

return x;

}

}

int Stack::peek()

{

if (top < 0) {

cout << "Stack is Empty";

return 0;

}

else {

int x = a[top];

return x;

}

}

bool Stack::isEmpty()

{

return (top < 0);

}

int main()

{

class Stack s;

s.push(10);

s.push(20);

s.push(30);

cout << s.pop() << " Popped from stack\n";

//print all elements in stack :

cout<<"Elements present in stack : ";

while(!s.isEmpty())

{

// print top element in stack

cout<<s.peek()<<" ";

// remove top element in stack

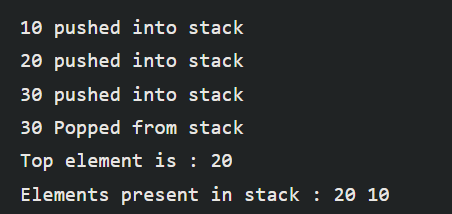
s.pop();

}

return 0;

}

**Output:**

****

**Result:** Hence, stack is implemented.