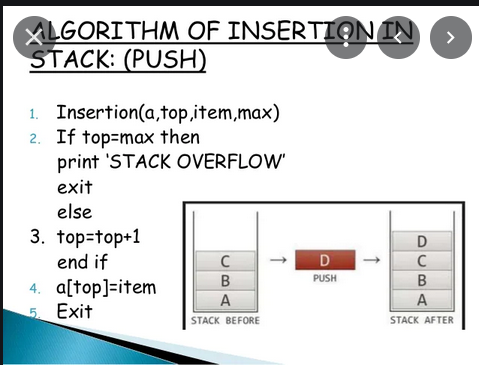
1. push, pop, display  
=>   
#include <iostream>  
#include  <conio.h>  
using namespace std;  
  
#include <iostream>  
using namespace std;  
int stack[100], n=100, top=-1;  
void push(int val) {  
   if(top>=n-1)  
   cout<<"Stack Overflow"<<endl;  
   else {  
      top++;  
      stack[top]=val;  
   }  
}  
void pop() {  
   if(top<=-1)  
   cout<<"Stack Underflow"<<endl;  
   else {  
      cout<<"The popped element is "<< stack[top] <<endl;  
      top--;  
   }  
}  
void display() {  
   if(top>=0) {  
      cout<<"Stack elements are:";  
      for(int i=top; i>=0; i--)  
      cout<<stack[i]<<" ";  
      cout<<endl;  
   } else  
   cout<<"Stack is empty";  
}  
int main() {  
   int ch, val;  
   cout<<"1) Push in stack"<<endl;  
   cout<<"2) Pop from stack"<<endl;  
   cout<<"3) Display stack"<<endl;  
   cout<<"4) Exit"<<endl;  
   do {  
      cout<<"Enter choice: "<<endl;  
      cin>>ch;  
      switch(ch) {  
         case 1: {  
            cout<<"Enter value to be pushed:"<<endl;  
            cin>>val;  
            push(val);  
            break;  
         }  
         case 2: {  
            pop();  
            break;  
         }  
         case 3: {  
            display();  
            break;  
         }  
         case 4: {  
            cout<<"Exit"<<endl;  
            break;  
         }  
         default: {  
            cout<<"Invalid Choice"<<endl;  
         }  
      }  
   }while(ch!=4);  
   return 0;  
}  
  
  
2.given by mam for push pop  
==>   
  
#include <iostream>  
  
using namespace std;  
  
int a[10],top;  
int isfull()  
{  
    if(top==10)  
        return 1;  
    else  
        return 0;  
}  
int  isempty()  
{  
    if(top==-1)  
        return 1;  
    else  
        return 0;  
}  
void push(int ele)  
{  
    if(isfull())  
        cout<<"stack overflow \n";  
    else  
        top++;  
        a[top]=ele;  
        cout<<ele<<"push \t";  
}  
void pop()  
{  
    if(isempty())  
        cout<<"stack underflow \n";  
    else  
        {  
            int data;  
            data=a[top];  
            top--;  
            cout<<data<<"pop out \t";  
        }  
}  
void disp()  
{  
    if(isempty())  
        cout<<"\nstack is empty \n";  
    else  
        for(int i=top;i>=0;i--)  
        {  
            cout<<endl;  
            cout<<a[i]<<"\t";  
            cout<<endl;  
        }  
}  
  
int main()  
{  
    top=-1;  
    push(11);  
    push(12);  
    push(13);  
    push(14);  
      
    disp();  
      
    pop();  
    pop();  
      
    disp();  
      
    pop();  
    pop();  
    disp();  
      
    pop();  
    pop();  
    disp();  
      
    return 0;  
      
      
      
}

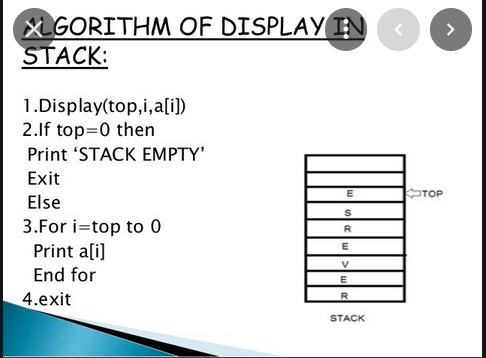
Push operation

* **Step 1** − Checks if the stack is full.
* **Step 2** − If the stack is full, produces an error and exit.
* **Step 3** − If the stack is not full, increments **top** to point next empty space.
* **Step 4** − Adds data element to the stack location, where top is pointing.
* **Step 5** − Returns success.

Pop operation

* **Step 1** − Checks if the stack is empty.
* **Step 2** − If the stack is empty, produces an error and exit.
* **Step 3** − If the stack is not empty, accesses the data element at which **top** is pointing.
* **Step 4** − Decreases the value of top by 1.
* **Step 5** − Returns success.





**How to push elements in stack using linked list**

Insertion of new element to stack is known as *push operation* in stack. We can push elements at top of stack.

Step by step descriptive logic to push elements in stack.

1. Check stack overflow, i.e. if (size >= CAPACITY), then print "Stack overflow" error message. Otherwise move to below step.
2. Create a new stack node using dynamic memory allocation i.e. struct stack \* newNode = (struct stack \*) malloc(sizeof(struct stack));.
3. Assign data to the newly created node using newNode->data = data;.
4. Link new node with the current stack top most element. Say newNode->next = top; and increment size count by 1.
5. Finally make sure the top of stack should always be the new node i.e. top = newNode;.

**How to pop elements from stack using linked list**

Removal of top most element from stack is known as *pop operation* in stack.

Step by step descriptive logic to pop elements from stack.

1. If size <= 0 then throw "Stack is Empty" error, otherwise move to below step.
2. Assign the top most element reference to some temporary variable, say struct stack \*topNode = top;. Similarly copy data of stack top element to some variable say int data = top->data;.
3. Make second element of stack as top element i.e. top = top->next;.
4. Delete the top most element from memory using free(topNode);.
5. Decrement stack size by one and return data.