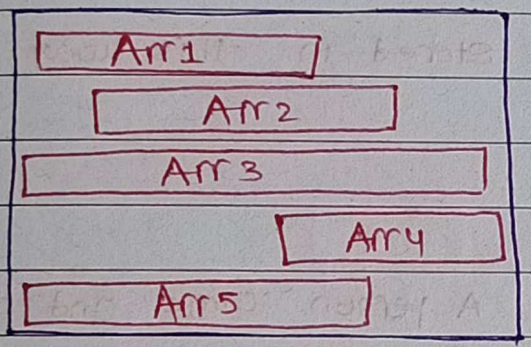
Lecture - 35

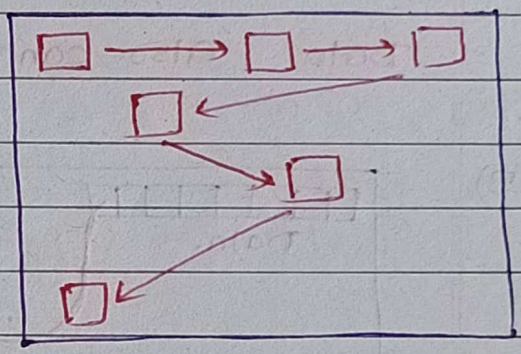
Stack

Till now, for data storage we can use either Array or Linked List.

For Array :



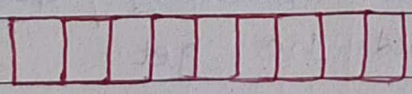
For linked list :



We have keep the data in two ways.
How to use this data ?

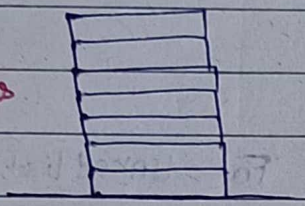
Let us understand with the help of real Example :

1) packet of Biscuit are arranged on a table -



If a person demand for biscuit, then he take either from front or back.

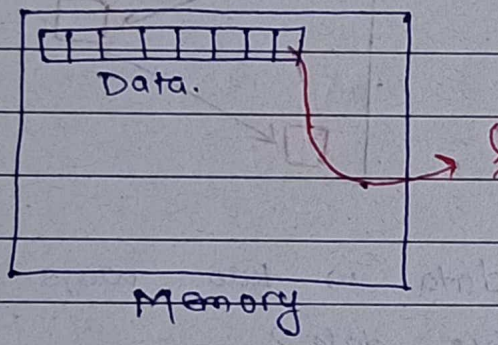
2) In a stationary shop, Copy is stored on table :-



A student demand for a copy then shopkeeper give you 2 copies from top.

Data also can be stored in this way.

3)



A person come and demand for some data. then from memory he get ~~from~~ from last.

If he demand more memory then the next memory he get.

What if he demand memory / data from Middle, then how he get.

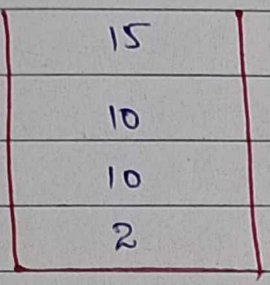
Here, We assume that he only want data from last.

We can also see the case we he get from first or middle.

Teacher's Signature.....

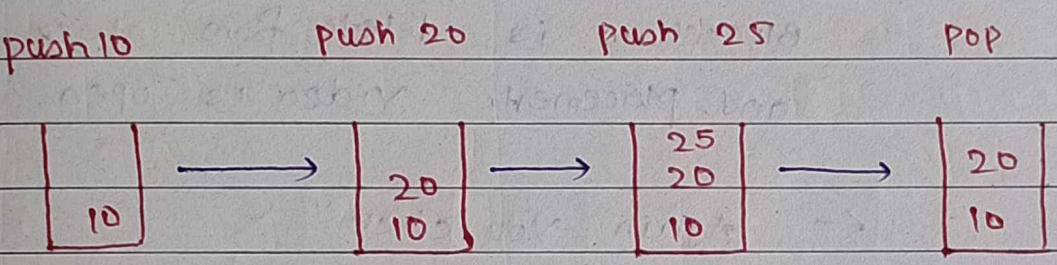
Page No.:
Date: / /

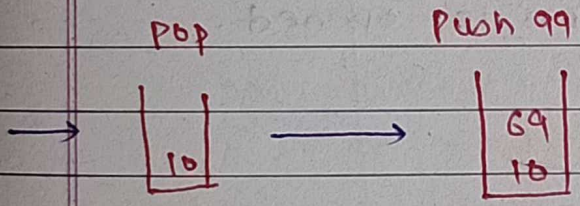
Stack (visual)



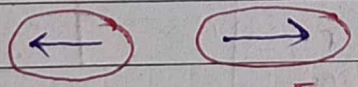
push : Insert an element
pop : Take the top element
top : Just Watch top element

How stack Mode?





In Web browser, you see :



back
undo

Forward
redo

Example:

on youtube you watch videos :

Teacher's Signature.....

Open Rohit Negi channel

4	Blockchain
3	placement
2	IIT Madras
1	Gate video

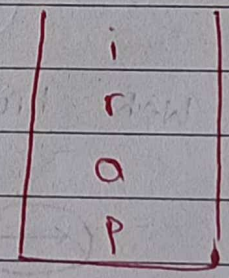
When you click this ←

Blockchain is delete from stack and placement video is open.

← Again you click placement video is deleted and IIT Madras is opened.

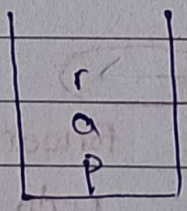
* undo operation (ctrl + z)

Write : Pari



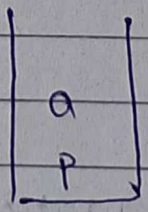
type ctrl + z

i deleted



type ctrl + z

r deleted

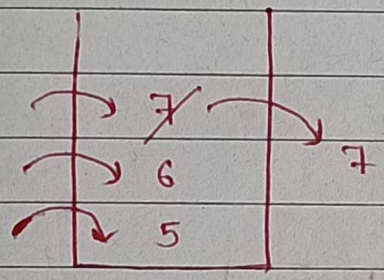


Teacher's Signature.....

* How to create a stack :

Push ()
Pop ()

LIFO (Last in first out)
↳ used in stack.



We can create a stack in two ways :

1) Creation by Array
2) Creation by Linked list

1) Creation By Array :

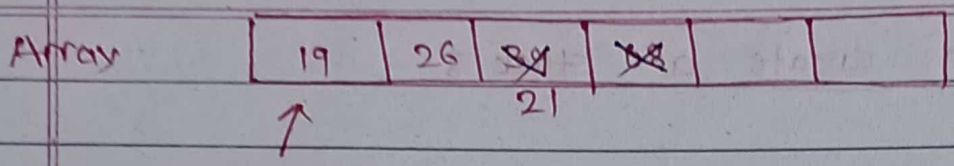
2	3	7	11	19	23

DrawBack :

Size is fixed. If we want to store more data, then it is not possible.

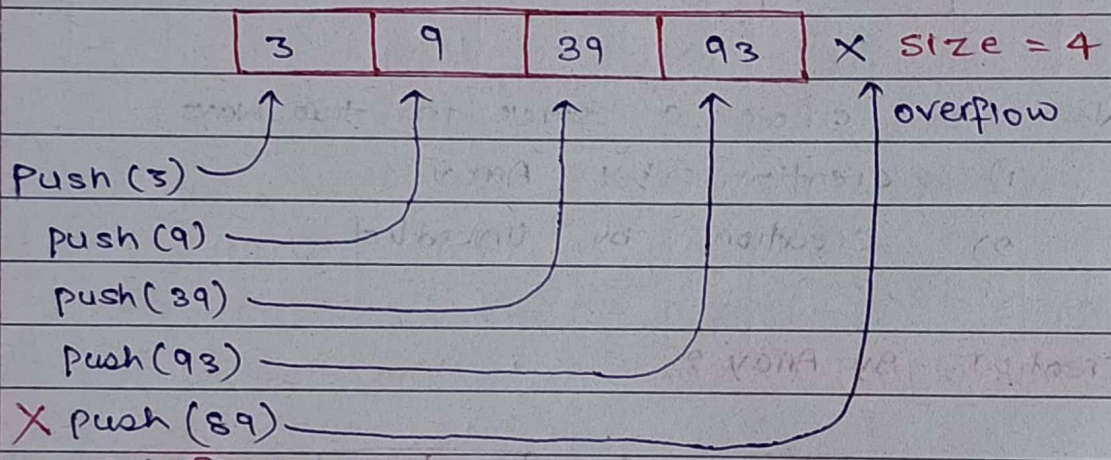
Array size = 6

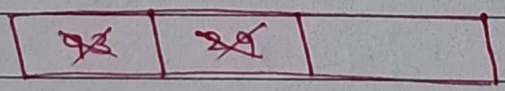
Let Make an array of size = 6



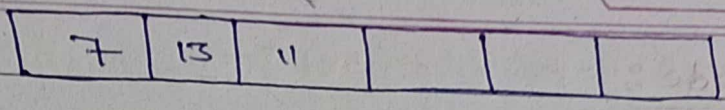
push (19)
push (26).
push (39)
push (08)
pop → delete 08
pop → delete 39
push (21)

* Overflow & underflow :

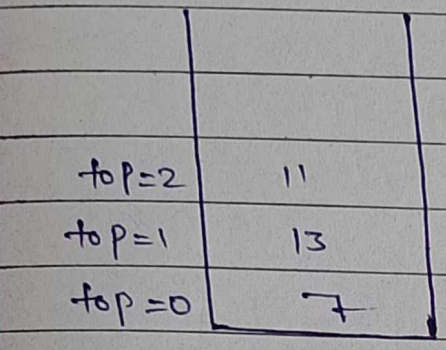




Push (93)
Push (29)
pop → delete 29
pop → delete 93
pop → underflow because No element present.

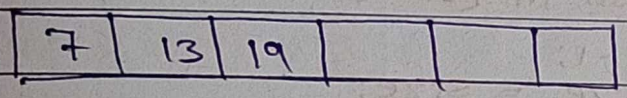
Arr → 

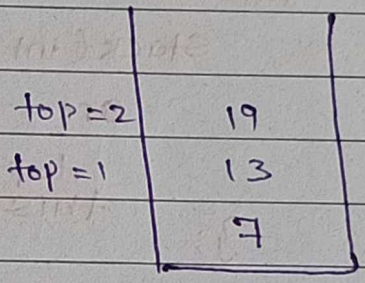
Initially top = -1



top = -1

① push 7 top++
② push 13 top++
③ push 11 top++
④ pop top--





⑤ push 19

* Condition for overflow & underflow :

(top == arr.size())
Stack overflow // push not possible

(top == -1)
Stack underflow // pop not possible.

S.top() → from this we can print the top element.

Page No.
Date : / /

Code :

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

class stack {
    int top;
    int * arr;
    int size;
  public:
    stack(int size) {
        arr = new int [size];
        top = -1;
        this->size = size;
    };

    void push( int data) {
        if (top == size-1) {
            cout << " stack overflow \n";
            return;
        }
        top++;
        arr[top] = data;
    }

    void pop() {
        if (top == -1) {
            cout << " underflow \n";
            return;
        }
        top--;
        return;
    }
```

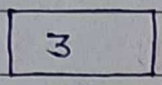
Teacher's Signature.....

```
int peak () {
    if (top == -1) {
        Cout << " stack underflow";
        return -1;
    }
    return arr[top];
}

bool empty () {
    return top == -1;
}

int main () {
    Stack S(4);
    S.push(4);
    S.push(19);
    S.push(413);
    Cout << S.peek();
    return 0;
}
```

Output : 413

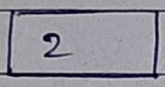
* Stack by Unicedlist :-

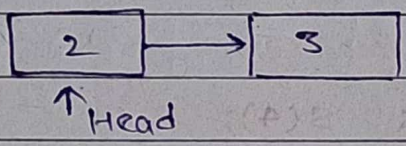
Push (3)



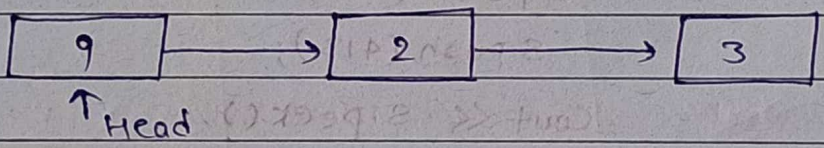
New Element आते जाएगा उसको Head बनाते जाएंगे।

push (2)

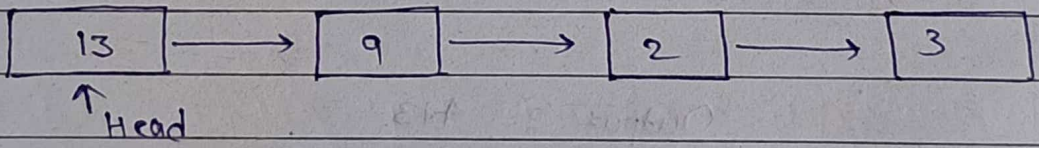




push (9)

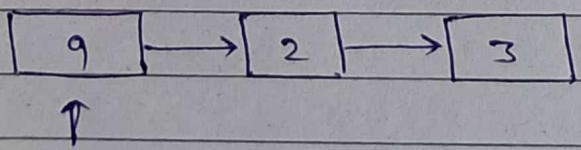


Push (13)



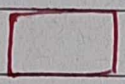
pop ()

Delete 13



push() → O(1) ✓
pop() → O(1) ✓
top() → O(1) ✓

Empty : Condition



If Head → NULL
then Empty

Where we use stack :
→ On Browser
→ For undo operation. ←
→ for Redo operation. →

Youtube → channel (Rohit Negi) → placement → 2 crore
→ Data → IIT Madras → Blockchain

Blockchain
IIT Madras
Gate
2 Crore
Placement

After this, if we press
← ↳ Delete the Blockchain
← ↳ Delete the IIT Madras
→ ↳ Again come IIT Madras.

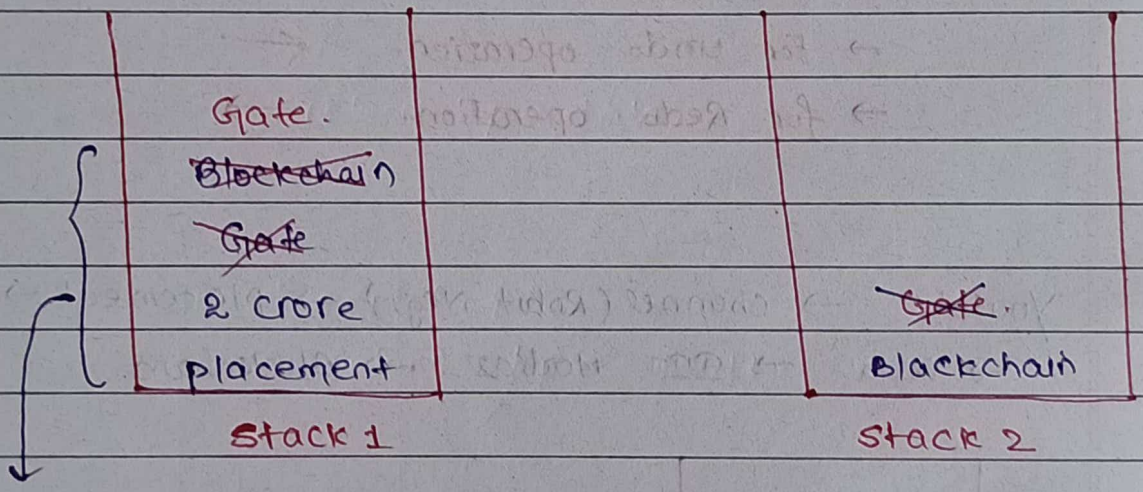
Teacher's Signature.....

After deleting How it Comes again by clicking this (→)

By the help of 2 stack.

When we undo (←) it is deleted from one stack and stored in another stack.

And, When we do Redo (→) it is again Come is 1st stack and deleted in 2nd stack.



After this if we put undo (←) it is deleted from stack 1 and stored in stack 2.

← (undo) Again we put undo, it delete Gate from stack 1 and stored in stack 2;

→ (Redo) If we put Redo, it delete from stack 2 and again Come to stack 1.

Teacher's Signature.....

Page No.:
Date: / /

Code :

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

class Node {
    public :
        int data;
        Node * next;
        Node (int data) {
            this -> data = data;
            next = NULL;
        }
};

class Stack {
        Node * top;
        public :
            Stack () {
                top = NULL;
            }

    Void push (int data) {
            Node * temp = new Node (data);
            if (!temp) {
                Cout << "stack overflow \n";
                return;
            }
            temp -> next = top;
            top = temp;
        }
```

Teacher's Signature.....

```
Void pop() {
    if (! top) {
        Cout << " stack underflow \n";
            return;
        }
        Node * temp = top;
        top = top → next;
        delete temp;
    }

int peek() {
    if (!top) {
        Cout << " Stack is empty \n" ;
        };
        return top → data;
    };

    bool empty() {
        return top == NULL;
    };
};

int main() {
    Stack s;
    s.push(10);
    s.push(20);
    Cout << s.empty();
    return 0;
}
```

Teacher's Signature.....

Problem : 1

Stack : function call

* Stack with help of STL

```
stack<int> s;
stack<char> s;
s.push(4);
s.pop();
s.push(9);
```

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

int main() {
    stack<int> s;
    s.push(14);
    s.push(19);
    s.pop();
    cout << s.empty();
    return 0;
}
```

s.size()

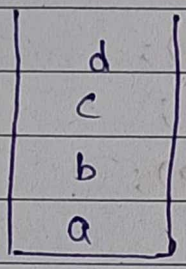
Page No.:
Date: / /

* problem - 1 : (Not from GFG)

String : "abcd"

Reverse this by stack

put string in stack



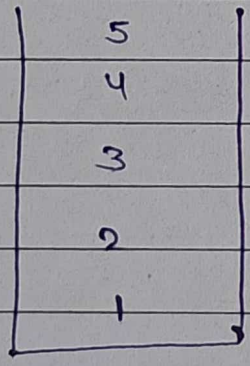
Now pop one by one and print

"dcba"

result.

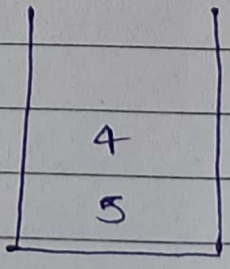
* problem - 2 :- (delete Middle element of a stack)

GFG



Delete Mid element

① Copy n/2 element in another stack



Teacher's Signature.....

(2) pop next element

→ 3

(3) Then, return the element from stack 2

5.
4
2
1

O(N)

If length = even

then put n/2 element in another stack.

If length = odd

then put n/2 element in another stack.

Code:

```
Void deleteMid ( Stack <int> &s, int size) {
        Stack <int> temp;
        int Count = size / 2;
        While ( Count --) {
            temp.push ( s.top());
            s.pop();
        };
        s.pop();
        While(temp.size()) {
            s.push (temp.top());
            temp.pop();
        }
}
```

Teacher's Signature.....

GfG Problem : 3 (parenthesis checker 2744)

Given Expression string x.

String can consists only
{ } [ ] ( )

Check that parenthesis is correctly placed or not.

[( )] → balanced

{ } → balanced

([) → Not balanced / invalid.

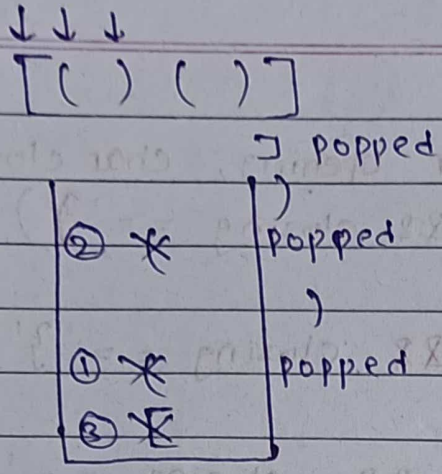
How to check :

[( ) ( )] → valid

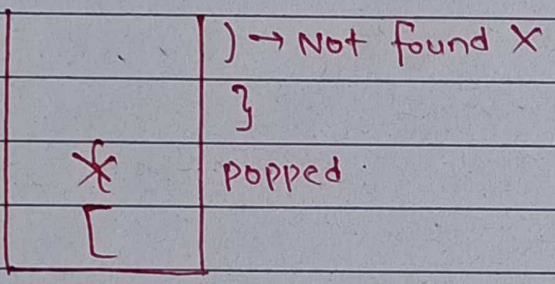
If we met an open, then we find for closed, if closed found then valid otherwise invalid.

If we met an close before open
↳ then it is invalid..

Teacher's Signature.....



[{ })



Invalid

Code :

```
bool ispar (string exp) {
    stack <char> s;
    for (int i = 0; i<exp.length(); i++) {
        if (exp[i]=='(' || exp[i]=='{' || exp[i]=='[')
            s.push(exp[i]);
        else if(exp[i]==')' || exp[i]=='}' || exp[i]==']')
        {
            if (s.empty() || !isMatchingpair(s.top(), exp[i])
                return false;
            else
                s.pop();
        }
    }
    return s.empty();
}
```

Teacher's Signature.....

~~bool is M~~

```
bool isMatchingpair(char opening, char closing){
    if(opening == '(' && closing == ')')
        return true;
    else if(opening == '{' && closing == '}')
        return true;
    else if(opening == '[' && closing == ']')
        return true;
    return false;
}
```

Teacher's Signature.....