Stack

→ an ordered list

→ container following a rule

for – insertion

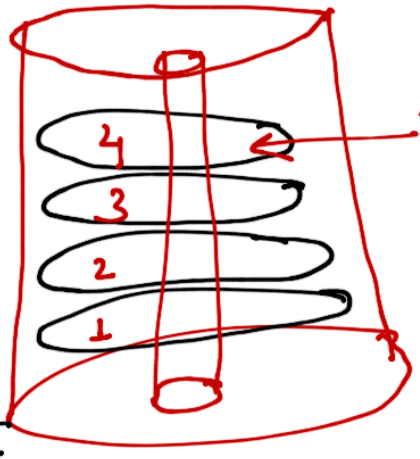
– deletion

→ Based on

LIFO (Last In First Out)

or

FILO (First In Last Out)



CD - Bag

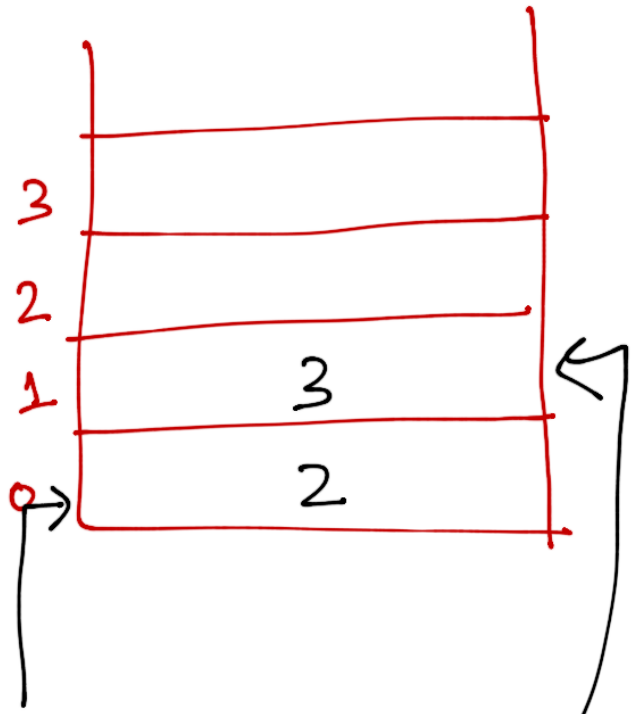
Operations

- Push (Insert an element)
- Pop (Remove an element)
- Peek (Gives the top element
- isEmpty
- isFull

Size = 4

At start : top = -1

pop()
↓
underflow
condition



Push(2) ⟶ top++;

Push(3) ⟶ top++;

size = 4

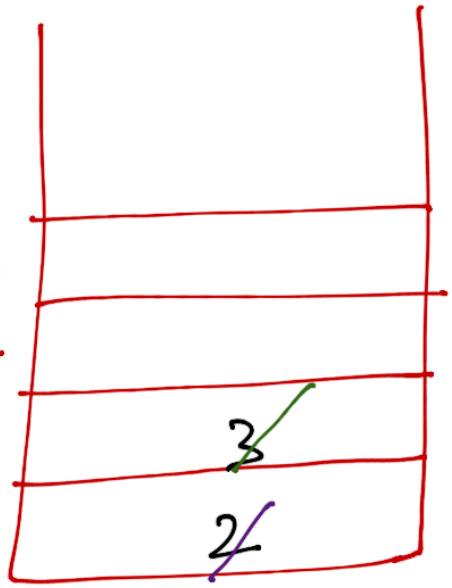
Pop() → Top--;

Pop() → Top--;

Top = -1

Top → 1

Top → 0



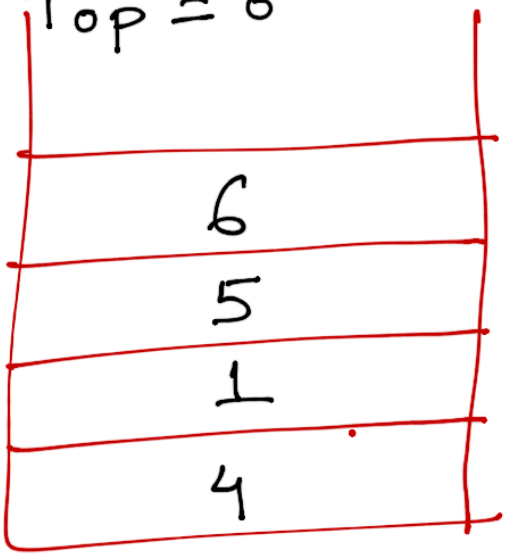
Push(4) → Top++ | Top = 0

Push(1) → Top++ | Top=1

Push(5) → Top++ | Top=2

Push(6) → Top++ | Top=3

Top → 0



Push(7) → Overflow

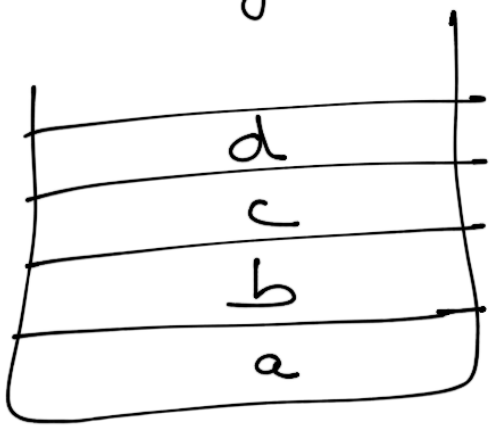
isFull() → True

isEmpty() → False

Applications

1. Reverse a string

abcd → dcba



2. Undo in Text Editor

abcde

←

3. Recursion / Function Call

A function calling itself

4. Balance of Paranthesis

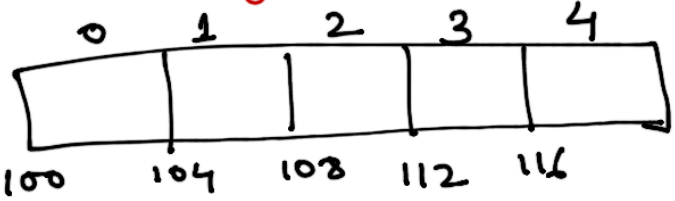
{
{
{
}
}
}

5. Infix to Postfix / Prefix

6. DFS / Tower of Hanoi

7. Evaluation of Postfix Exp.

Implementation Using Arrays

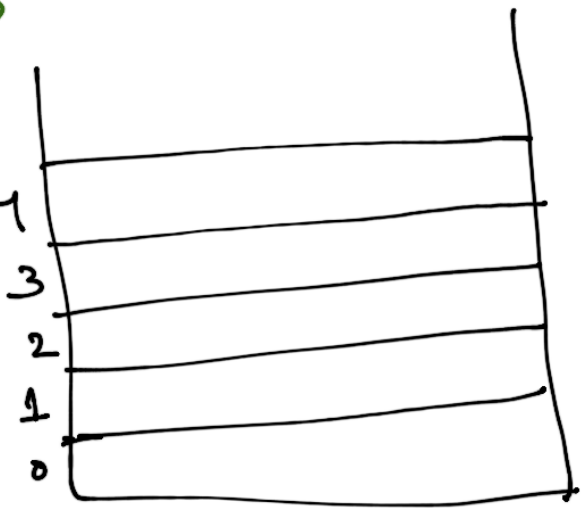


int a[5];

Memory = 5 x 4

= 20 bytes

Rotate



int stack[5]

1. PeeK()

```
begin procedure peeK
    return stack[top]
end procedure
```

Operations

push()
pop()
peeK()
display()
isFull()
isEmpty()

2. isfull()

int stack[MAXSIZE

```
begin procedure isfull
    if top equals to MAXSIZE
        return true
    else
        return false
    endif
end procedure
```

3. isEmpty()

```
begin procedure isEmpty
    if top less than 1
        return true
    else
        return false
    endif
end procedure
```

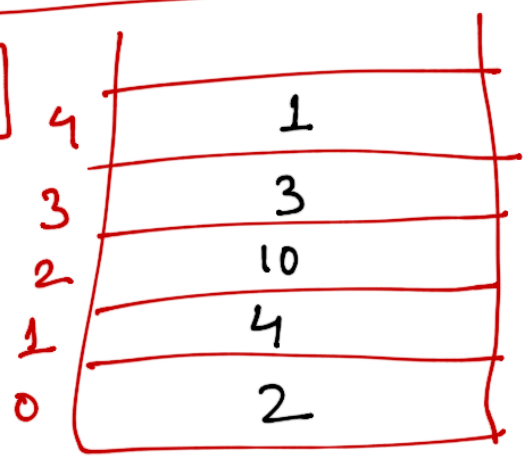
4. Push()

```
begin procedure push: stack, data
    if stack is full
        return null  ← Stack Overflow.
    endif

    top ← top + 1
    Stack [top] ← data

end procedure
```

```
                    int stack[size]
void push(int data)
{   if (top == size-1)
        { print ("Overflow")
    else }
        { top++ ;
        } stack[top] = data;
}
```



5. Pop()

```
begin procedure pop: stack
    if stack is empty
        return null        ← Stack Underflow
    endif
    data ← stack[top]
    top ← top - 1
    return data
end procedure
```

```
void pop()
{  int item;
    if (top == -1)
    {   print ("Underflow");
    }
    else
    {   item = stack[top];
        top--;
}
```

All operations

```
do
{
Enter choice 'ch' : 1. Push, 2. Pop, 3. Peek,
                    4. Display

  Switch (ch) :

        Case 1 : Push();
                 break;
        Case 2 : Pop();
                 break;
        Case 3 : Peek;
                   break;
        Case 4 : display()
                   break;
        default : Invalid choice

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