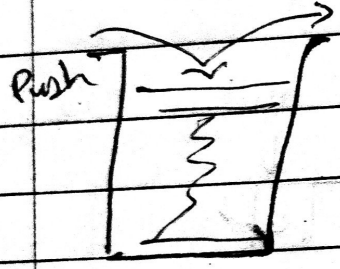
CHAPTER

STACKS

→ A stack is a non-primitive linear data structure.
→ addition and deletion of data is done from only one end. Known as top of stack (TOS).
→ Follows Last In First out (LIFO)



★ Stack implementation :-

Stack can be implemented in two ways :-

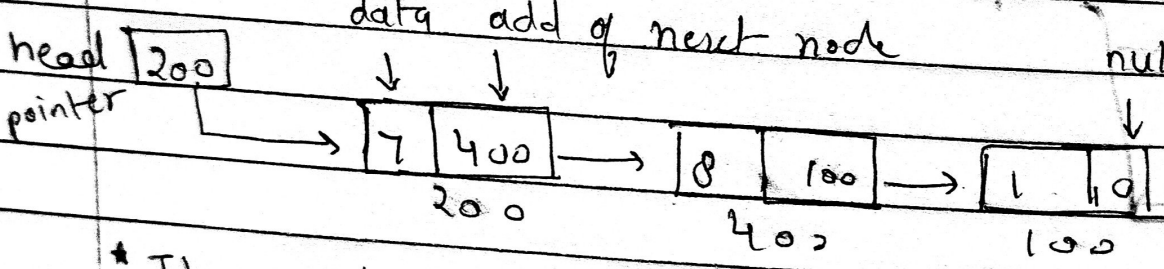
i) Static implementation
ii) Dynamic implementation

(i) Static implementation :- Uses arrays to create stack. It is a simple technique but not flexible way of creation.

ii) Dynamic implementation :- It is also called linked list representation and uses pointers to implement the stack type of data structure.

linked list implementation of stack :-

Singly linked list :-

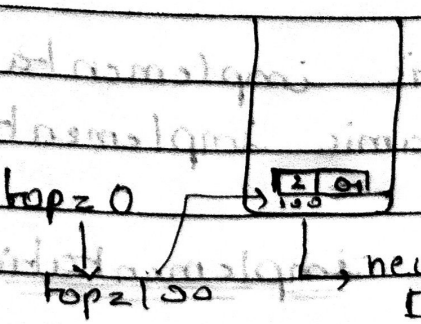


* It must follows the concept of LIFO
* Space complexity O(1) & time complexity O(n)
* We are taking head as top

Eg →

i) Push(2)

→ storing 0 in a node [2|0]

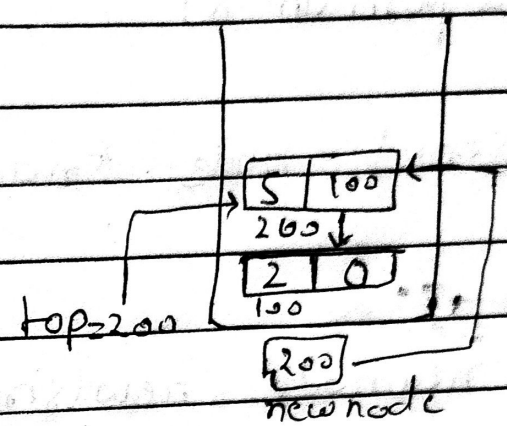


→ Now top should point to top node address
So top → top=100

2) Push(5)

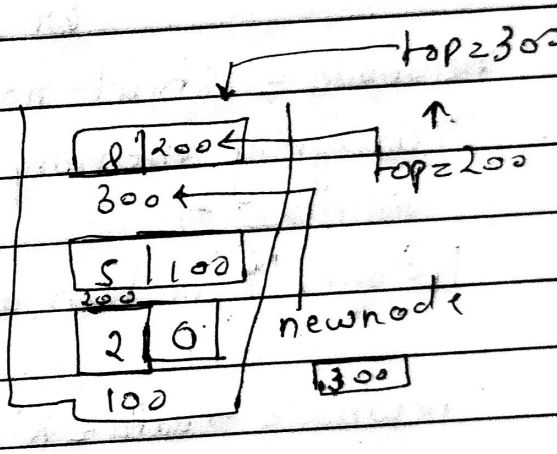
Now we push (5)
it contain address of
previous node (from top)

top will now point (5)



3) Push (8)

Vertically we are
creating linked list.



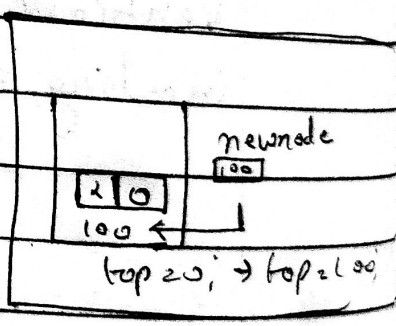
// Code to implement stack

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

struct node {
    int data;
    struct node *link; // next part, pointer part for todd
};
struct node *top = 0; // data type of struct node
                      Which *top pointer point
                      to.
                      (currently it is not pointing
```

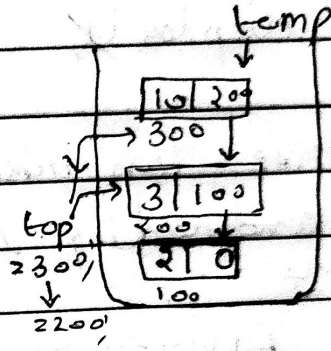
```
void push (int x)
{
  struct node *newnode; // creating node pointing to the node
  // newnode -> new(struct node); allocating memory to Pointer
                    or
  newnode = (struct node *) malloc (sizeof (struct node));
  /* assigning memory to newnode how much byte it will take */
  // 8 bytes have been allocated
  newnode -> data = x;
  newnode -> next = top;
  top = newnode; // we have 100 in newnode;
  // when the newnode is created it would have it address
}

void pop(){
  struct node *temp; // to free memory
  temp = top;
  if (top >= 0)
  { cout << " stack is already empty"; }
  // if top contains nothing
```



```
else
{
cout << " The data" << top->data << " will be removed";

top= top->link;   // top will point to next element
Free (temp);
    or
delete (temp);
}
}
```




```
void display ()
{
// have to create temp pointer
struct node *temp;
temp = top;
while (temp!=0)   // while (temp->link!=0)
  {
    cout << temp->data;
    temp= temp->link;
  }
}
cout << temp->data;  // last element will not be
                        printed
```

```
//Driver code
void main()
{
push(3);
Push(4);
display();
```

//we can take input by changing function variable and adding some lines

}

APPLICATIONS OF STACKS :-

★ STACK frames :- A stack frame is a memory management technique used in some programming languages for generating and eliminating temporary variables.

2) Reversing a String :- As the characteristic of stack is reversing the order of execution. It is useful in reversing the string values.