**Array Implementation of Stack:**

#include <stdio.h>

int A[10], top=-1; // Creating Stack

int size\_A = sizeof(A)/sizeof(A[0]); // Calculating Size of Array

/\* Driver Functions \*/

void Push(int data);

int Pop();

void PrintStack();

int IsEmpty();

/\* Main Method \*/

int main()

{

Push(0);

Push(1);

Push(2);

Push(3);

Pop(); // Delete 3 from starting

Pop(); // Delete 2 from starting

PrintStack(); //Print Stack

return 0;

}

/\* Check for empty stack \*/

int IsEmpty()

{

if(top==-1)

return 1;

else

return 0;

}

/\* Insert Element \*/

void Push(int data)

{

if(IsEmpty())

top = 0;

else if( top == size\_A )

printf("\n Stack is Full \n");

else

top++;

A[top] = data;

}

/\* Delete Element \*/

int Pop()

{

if(IsEmpty())

printf("\n Stack is Empty \n");

else

top--;

return A[top-1];

}

/\* Print Stack \*/

void PrintStack()

{

int i=0;

for(i=0;i<=top;i++)

printf(" %d",A[i]);

}

List Implementation of Stack:

#include <stdio.h>

#include <stdlib.h>

/\* List Structure \*/

typedef struct Node

{

int data;

struct Node \*link;

}node;

int NoOfNodes=0; // Count number of node

node \*head=NULL; // Head node to keep track of list

/\* Driver Functions \*/

void Push(int data);

int Pop();

void PrintStack(node \*);

/\* Main Method \*/

int main()

{

Push(0);

Push(1);

Push(2);

Push(3);

Pop(); // Delete 3 from Stack

Pop(); // Delete 2 from Stack

PrintStack(head); // Print Stack data

return 0;

}

/\* Insert Element \*/

void Push(int data)

{

// Declaring node

NoOfNodes++;

node \*temp = (node\*)calloc(1,sizeof(node));

temp->data = data;

temp->link = NULL;

// If head is NULL or first node

if(!head)

{

head = temp;

return;

}

// Traverse list upto end

node \*traverse=head;

while(traverse->link)

traverse = traverse->link;

traverse->link = temp;

}

/\* Delete Element \*/

int Pop()

{

node \*traverse = head;

for(int i=0;i<NoOfNodes-2;i++)

traverse = traverse->link;

node \*Delete = traverse->link;

traverse->link = NULL;

int data = Delete->data;

free(Delete);

NoOfNodes--;

return data;

}

/\* Print Stack \*/

void PrintStack(node \*p)

{

printf(" %d",p->data);

if(!p->link)return;

PrintStack(p->link);

}