STACK IMPLEMENTATION

AIM :- Write a program to implement a STACK data structure by

[A] Array [B] Linked list

Demonstrate the STACK operation such as push, pop and print using menu driver .

PROGRAM :-

[A] By using ARRAY :-

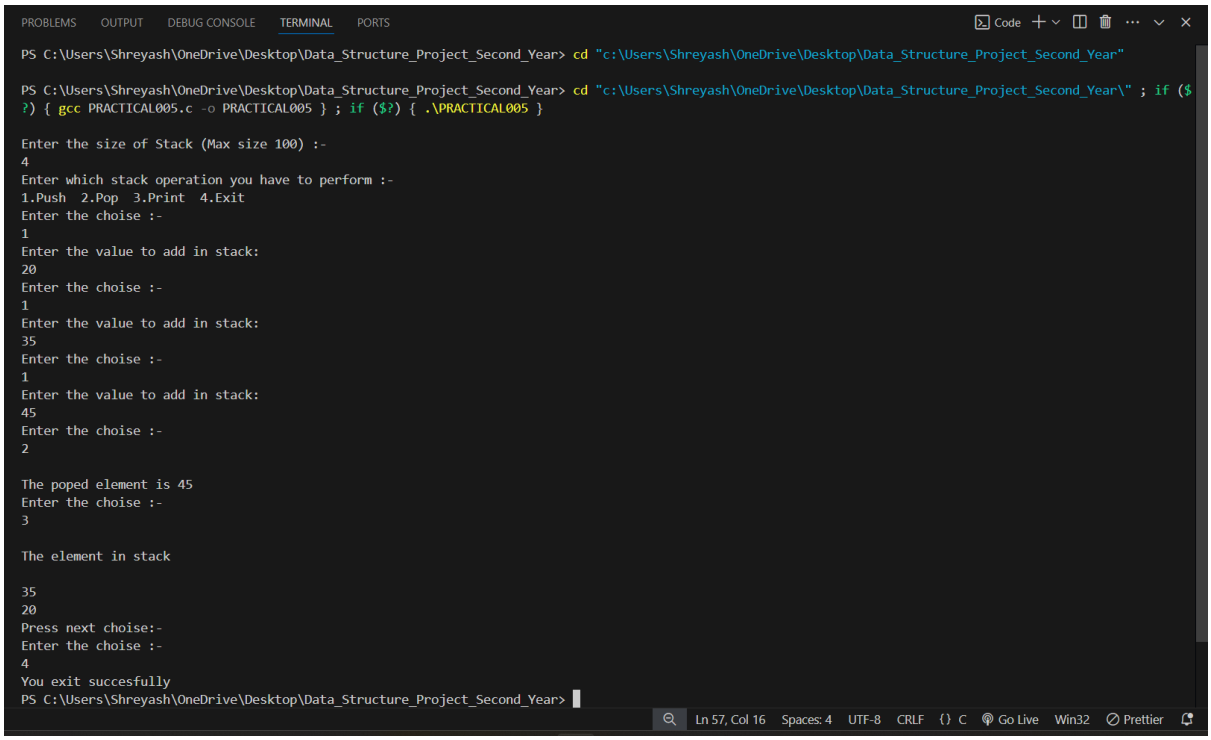
```
#include <stdio.h>
// Stack implementation using array
int stack[100], top, n, x;

// Function for PUSH :-
void push()
{
   if (top >= n - 1)
   {
      printf("\nStack is overflow\n");
   }
   else
   {
      printf("Enter the value to add in stack: \n");
      scanf("%d", &x);
      top++;
      stack[top] = x;
   }
}

// Function for POP :-
void pop()
{
   if (top <= -1)
   {
```

```
        printf("\nStack is Underflow\n");
    }
    else
    {
        printf("\nThe poped element is %d\n", stack[top]);
        top--;
    }
}

// Function for printing the element in stack :-
void print()
{
    if (top >= 0)
    {
        printf("\nThe element in stack\n");
        for (int i = top; i >= 0; i--)
        {
            printf("\n%d", stack[i]);
        }
        printf("\nPress next choise:-\n");
    }
    else
    {
        printf("\nThe Stack is empty.\n");
    }
}

int main()
{
    int choise;
    top = -1;
```

```
printf("Enter the size of Stack (Max size 100) :-\n");
scanf("%d", &n);
printf("Enter which stack operation you have to perform :-\n");
printf("1.Push  2.Pop  3.Print  4.Exit\n");
do
{
   printf("Enter the choise :-\n");
   scanf("%d", &choise);

   switch (choise)
   {
   case 1:
      push();
      break;

   case 2:
      pop();
      break;

   case 3:
      print();
      break;

   case 4:
      printf("You exit succesfully\n");
      break;

   default:
      printf("Please enter a valid number!!\n");
      break;
   }
```

```
    } while (choise != 4);


    return 0;
}
```

OUTPUT



[B] By using LINKED LIST :-

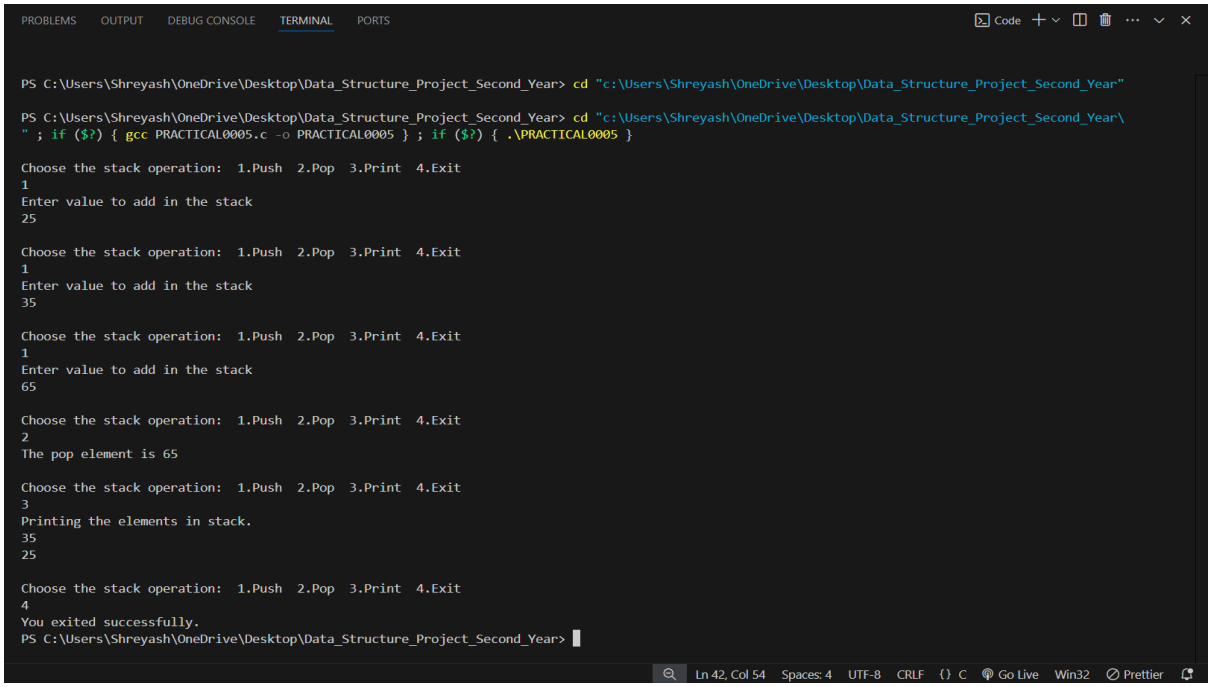
```
#include <stdio.h>
#include <stdlib.h>

// Stack implementation using linked list :-

//Blueprint for the node (structure) :-
struct node
{
   int data;
   struct node *next;
};
struct node *Top = NULL;

//Function for entering the element into the stack :-
void push(int x)
{
   struct node *new;
   new = (struct node *)malloc(sizeof(struct node));

   if (new == NULL)
   {
      printf("Stack overflow\n");
      return;
   }
   new->data = x;
   new->next = Top;
   Top = new;
}
```

```
//Function for pop (deleting) the element in stack :-
void pop()
{
   struct node *t;
   if (Top == NULL)
   {
      printf("Stack underflow\n");
   }
   else
   {
      t = Top;
      Top = Top->next;
      free(t);
      t = NULL;
   }
}

//Function to print the element in the stack :-
void print()
{
   struct node *temp = Top;
   if (Top == NULL)
   {
      printf("Stack is empty.\n");
      return;
   }
   printf("Printing the elements in stack.\n");
   while (temp != NULL)
   {
      printf("%d\n", temp->data);
      temp = temp->next;
```

```
    }
}

int main()
{
    int choice;
    int add;

    do
    {
        printf("\nChoose the stack operation:  1) Push  2) Pop  3) Print  4) Exit \n");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter value to add in the stack\n");
            scanf("%d", &add);
            push(add);
            break;
        case 2:
            pop();
            break;
        case 3:
            print();
            break;
        case 4:
            printf("You exited successfully.\n");
            break;
        default:
            printf("Please enter the valid choice!!\n");
```

```
        }
    } while (choice != 4);


    return 0;
}
```

OUTPUT



GIT-HUB LINK for practical :-

https://github.com/ShreyashGajbhiye453/Data-Structure-Practical-No.-01