Stacks using Arrays:

```
#include <stdio.h>
#define MAXSIZE 5
struct stack
{
   int stk[MAXSIZE];
   int top;
};
struct stack s;
void push(void);
int  pop(void);
void display(void);
 int main ()
{
   int choice;
   int option = 1;
   s.top = -1;
    printf ("STACK OPERATION\n");
   while(option)
   {
      printf ("----------------------------------------\n");
      printf ("      1    -->    PUSH              \n");
```

```
   printf ("      2    -->    POP              \n");
   printf ("      3    -->    DISPLAY              \n");
   printf ("      4    -->    EXIT          \n");
   printf ("----------------------------------------\n");
printf ("Enter your choice\n");
   scanf    ("%d", &choice);
   switch (choice)
   {
   case 1:
      push();
      break;
   case 2:
      pop();
      break;
   case 3:
      display();
      break;
   case 4:
      return 0;
   }
   fflush (stdin);
   printf ("Do you want to continue(Type 1(y) or 0(no)?\n");
   scanf    ("%d", &option);
```

```
    }
return 0;
}
/*  Function to add an element to the stack */
void push ()
{
    int element;
    if (s.top == (MAXSIZE - 1))
    {
        printf ("Stack is Full\n");
        return;
    }
    else
    {
        printf ("Enter the element to be pushed\n");
        scanf ("%d", &element);
        s.top = s.top + 1;
        s.stk[s.top] = element;
    }
    return;
}
/*  Function to delete an element from the stack */
int pop ()
```

```
{
   int element;
   if (s.top == - 1)
   {
      printf ("Stack is Empty\n");
      return (s.top);
   }
   else
   {
      element = s.stk[s.top];
      printf ("poped element is = %dn", s.stk[s.top]);
      s.top = s.top - 1;
   }
   return(element);
}
/*  Function to display the status of the stack */
void display ()
{
   int i;
   if (s.top == -1)
   {
      printf ("Stack is empty\n");
      return;
```

```
    }
    else
    {
        printf ("\n The status of the stack is \n");
        for (i = s.top; i >= 0; i--)
        {
            printf ("%d\n", s.stk[i]);
        }
    }
    printf ("\n");
}
```

Stacks using single linked list :

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int info;
    struct node *ptr;
}*top,*top1,*temp;
 int topelement();
void push(int data);
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();
 int count = 0;
 int main()
{
    int no, ch, e;
    printf("\n 1 - Push");
    printf("\n 2 - Pop");
```

```
printf("\n 3 - Top");
printf("\n 4 - Empty");
printf("\n 5 - Stack Count");
printf("\n 6 - Dipslay");
printf("\n 7 - Destroy stack");
printf("\n 8 - Exit");
create();

while (1)
{
    printf("\n Enter choice : ");
    scanf("%d", &ch);
   switch (ch)
    {
    case 1:
        printf("Enter data : ");
        scanf("%d", &no);
        push(no);
        break;
    case 2:
        pop();
        break;
    case 3:
```

```
    if (top == NULL)
        printf("No elements in stack");
    else
    {
        e = topelement();
        printf("\n Top element : %d", e);
    }
    break;
case 4:
    empty();
    break;
case 5:
   stack_count();
    break;
case 6:
    display();
    break;
case 7:
    destroy();
    break;
case 8:
    exit(0);
default :
```

```
            printf(" Wrong choice, Please enter correct choice  ");
            break;
        }
    }
return 0;
}


/* Create empty stack */
void create()
{
    top = NULL;
}
 /* Count stack elements */
void stack_count()
{
    printf("\n No. of elements in stack : %d", count);
}
 /* Push data into stack */
void push(int data)
{
    if (top == NULL)
    {
```

```
        top =(struct node *)malloc(1*sizeof(struct node));
        top->ptr = NULL;
        top->info = data;
    }
    else
    {
        temp =(struct node *)malloc(1*sizeof(struct node));
        temp->ptr = top;
        temp->info = data;
        top = temp;
    }
    count++;
}

/* Display stack elements */
void display()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("Stack is empty");
        return;
```

```
    }

    while (top1 != NULL)
    {
        printf("%d ", top1->info);
        top1 = top1->ptr;
    }
 }

/* Pop Operation on stack */
void pop()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("\n Error : Trying to pop from empty stack");
        return;
    }
    else
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
    free(top);
```

```
    top = top1;
    count--;
}

/* Return top element */
int topelement()
{
    return(top->info);
}

/* Check if stack is empty or not */
void empty()
{
    if (top == NULL)
        printf("\n Stack is empty");
    else
        printf("\n Stack is not empty with %d elements", count);
}

/* Destroy entire stack */
void destroy()
{
    top1 = top;
```

```
    while (top1 != NULL)
    {
        top1 = top->ptr;
        free(top);
        top = top1;
        top1 = top1->ptr;
    }
    free(top1);
    top = NULL;

    printf("\n All stack elements destroyed");
    count = 0;
}
```