NATIONAL INSTITUTE OF TECHNOLOGY PUDUCHERRY
(An Institution of National Importance under MHRD, Govt. of India)
KARAIKAL – 609 609

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Roll Number: CS19B1009 **Name** ARUN KUMAR R

Semester: 2nd Semester **Class:** Computer Science and Engineering

Subject Code: CS106 **Subject Name:** Data Structures Laboratory

Exercise Number – 4

Date: 18.12.19

1.STACK

AIM:

To store the data in a memory location using stack algorithm.

ALGORITHM:

1. Start the program .
2. declare the variable.
3. Implement the stack function - push , peep, pop.
4. get the input from user.
5. manipulate the data from user with appropriate functions.
6. output the result.
7. end the program.

PROGRAM:

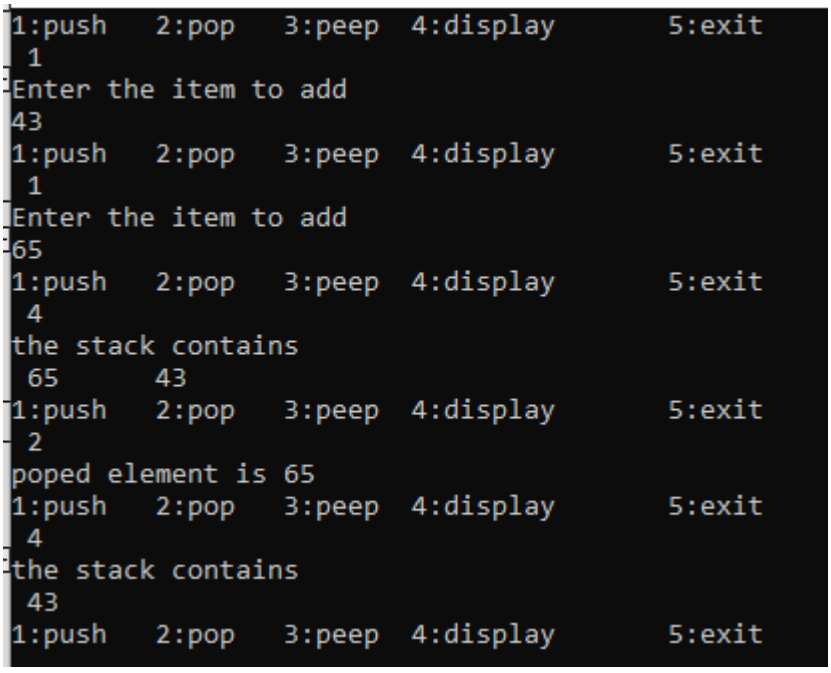
```
#include<stdio.h>
#include<stdlib.h>

#define SIZE 10

struct stack
{
        int items[SIZE];
        int top;
};

typedef struct stack stk;
int push(stk *s, int k)
{
        if(s->top == SIZE-1)
        {
                printf("stack overflow\n");
```

```
                return 0;
        }
        else{
                s -> top++;
                s -> items[s -> top] = k;
                return 1;
        }
}

int pop(stk *s)
{
        int k;
        if( s-> top == -1)
        {
                printf("STACK UNDERFLOW\n");
                return -1;
        }
        else
        {
                k = s -> items[s ->top];
                s -> top--;
                return k;
        }
}

int peep( stk * s)
{
        int k;
        if(s -> top == -1)
        {
                printf("stack underflow\n");
                return -1;
        }
        else
        {
                k = s ->items[s -> top];
                return k;
        }
}

int display(stk * s)
{
        int i = s->top;
         if(s -> top == -1)
         {
                printf("stack is empty\n");
                return 0;
         }
```

```
                printf("the stack contains\n");
        while(i > -1)
        {
                printf(" %d\t", s ->items[i]);
                i--;
        }
        printf("\n");
        return 0;
}
int main()
{
        int i, k;
        stk s;
        s.top = -1;
        while (1)
        {
                printf("1:push\t 2:pop\t 3:peep\t 4:display\t 5:exit\n ");
                scanf("%d", &i);
                switch(i)
                {
                        case 1:
                                {
                                        printf("Enter the item to add\n");
                                        scanf("%d",&k);
                                        push(&s, k);
                                        break;
                                }
                   case 2:
                                {
                                        k = pop(&s);
                                        if(k != -1)
                                        printf("poped element is %d\n", k);
                                        break;
                                }
                        case 3:
                                {
                                        k = peep(&s);
                                        if( k != -1)
                                        printf("the peeped item is %d\n", k);
                                        break;
                                }
                        case 4 :
                                display(&s);
                                break;

                        case 5 :
                        exit(0);
```

```
                default :
                {
                        printf("invalid input\n");
                        break;
                }
        }
    }
}
```

OUTPUT:



RESULT:

The program was executed successfully.

2.QUEUE

DATE:18.12.19

AIM:

To store given data in memory location using queue data structure

ALGORITHM:

1. Start the program.
2. declare the variable.
3. Implement the queue functions - enqueue, dequeue, peep.
4. get the input from the user.
5. store the data in queue data structure.
6. output the result.
7. end the program.

PROGRAM:

```
#include<stdio.h>

#define MAX 10

struct queue
{
   int a[MAX];
   int rear,front,count;
};

typedef struct queue Q;

int isEmpty(Q *q){
   if (q->count == 0)
   {
      return 1;
   }
   else
   {
      return 0;
   }
}


int isFull(Q *q){
   if (q->count == MAX)
   {
      return 1;
   }
```

```
    else
    {
        return 0;
    }
}

int enqueue(Q *q,int k){
    if (isFull(q))
    {
        printf("Queue is Full\n");
        return -1;
    }
    q->rear = (q->rear+1)%MAX;
    q->a[q->rear] = k;
    q->count++;
    if (q->count == 1)
    {
        q->front = q->rear;
    }
    printf("Enqueued\n");
}

int dequeue(Q *q){
    if (isEmpty(q))
    {
        printf("Queue is empty\n");
        return -1;
    }
    int k = q->a[q->front];
    q->front = (q->front + 1)%MAX;
    q->count--;
    if (q->count == 0)
    {
        q->rear = q->front = -1;
    }
    printf("Dequeued");
    return k;
}

int display(Q *q){
    int i = q->front;
    int j = q->count;
    if (isEmpty(q))
    {
        printf("Queue is Empty\n");
        return -1;
    }
```

```
    printf("Queue contains : \n");
    while (j)
    {
        printf("%d\n",q->a[i]);
        i++;
        j--;
    }

}
int  main(){
    Q q;
    q.rear = q.front = -1;
    q.count = 0;

    while (1)
    {
        int i,k;
        printf("Enter 1.enqueue 2.dequeue 3.display 4.exit\n");
        scanf("%d",&i);

        switch (i)
        {
        case 1:
        printf("Enter key : ");
        scanf("%d",&k);
            enqueue(&q,k);
            break;
        case 2:
            dequeue(&q);
            break;
        case 3:
            display(&q);
            break;
        case 4:
            return 0;
        default:
            break;
        }
    }
}
```

OUTPUT:

```
Enter 1.enqueue 2.dequeue 3.display 4.exit
1
Enter key : 54
Enqueued
Enter 1.enqueue 2.dequeue 3.display 4.exit
1
Enter key : 43
Enqueued
Enter 1.enqueue 2.dequeue 3.display 4.exit
3
Queue contains :
54
43
Enter 1.enqueue 2.dequeue 3.display 4.exit
2
DequeuedEnter 1.enqueue 2.dequeue 3.display 4.exit
3
Queue contains :
43
Enter 1.enqueue 2.dequeue 3.display 4.exit
```

RESULT:

The program was executed successfully.