S.No: 20	Exp. Name: ***Write a C program to implement different Operations on Queue using Dynamic Array***	Date:2023-06-13

Aim:

Write a program to implement queue using **dynamic array**.

In this queue implementation has

1. a pointer 'queue' to a dynamically allocated array (used to hold the contents of the queue)
2. an integer 'maxSize' that holds the size of this array (i.e the maximum number of data that can be held in this array)
3. an integer 'front' which stores the array index of the first element in the queue
4. an integer 'rear' which stores the array index of the last element in the queue.

ID: 224G1A0560
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

```
Sample Input and Output:
        Enter the maximum size of the queue : 3
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 2
        Queue is underflow.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 3
        Queue is empty.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 1
        Enter element : 15
        Successfully inserted.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 1
        Enter element : 16
        Successfully inserted.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 1
        Enter element : 17
        Successfully inserted.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 1
        Enter element : 18
        Queue is overflow.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 3
        Elements in the queue : 15 16 17
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 2
        Deleted element = 15
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 2
        Deleted element = 16
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 3
        Elements in the queue : 17
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 2
        Deleted element = 17
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 3
        Queue is empty.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 2
        Queue is underflow.
        1.Enqueue 2.Dequeue 3.Display 4.Exit
        Enter your option : 4
```

Source Code:

QUsingDynamicArray.c

```
#include <conio.h>
#include <stdio.h>
int *queue;
```


ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

```
int front, rear;
int maxSize;
void initQueue()
{
   queue = (int *)malloc(maxSize*sizeof(int));
   front = -1;
   rear = -1;
}
void enqueue(int X)
{
   if(rear == maxSize-1)
   {
      printf("Queue is overflow.\n");
   }
   else
   {
      rear++;
      queue[rear] = X;
      printf("Successfully inserted.\n");
   }
if(front == -1)
{
   front++;
}
}
void dequeue()
{
   if(front == -1)
{
   printf("Queue is underflow.\n");
}
else
{
   printf("Deleted element = %d\n",*(queue+front));
   if(rear == front)
   {
      rear = front = -1;
   }
   else
   {
      front++;
   }
   }
}
void display()
{
   if(front == -1 && rear == -1)
   {
      printf("Queue is empty.\n");
   }
else
{
   printf("Elements in the queue : ");
   for(int i = front; i <= rear; i++)
   {
      printf("%d ",*(queue+i));
```

```
    }
    printf("\n");
    }
}
int main()
{
    int op, X;
    printf("Enter the maximum size of the queue : ");
    scanf("%d", &maxSize);
    initQueue();
    while(1)
    {
       printf("1.Enqueue 2.Dequeue 3.Display 4.Exit\n");
       printf("Enter your option : ");
       scanf("%d" , &op);
       switch(op)
       {
          case 1:
          printf("Enter element : ");
          scanf("%d",&X);
          enqueue(X);
          break;
          case 2:
          dequeue();
          break;
          case 3:
          display();
          break;
          case 4:
          exit(0);
       }
    }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the maximum size of the queue : 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Queue is empty. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 15
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 16
Successfully inserted. 1

```
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option :  1
Enter element :  17
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option :  1
Enter element :  18
Queue is overflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option :  3
Elements in the queue : 15 16 17  2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option :  2
Deleted element = 15 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option :  2
Deleted element = 16 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option :  3
Elements in the queue : 17  2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option :  2
Deleted element = 17 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option :  3
Queue is empty. 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option :  2
Queue is underflow. 4
1.Enqueue 2.Dequeue 3.Display 4.Exit 4
Enter your option :  4
```

Test Case - 2

User Output

```
Enter the maximum size of the queue :  2
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option :  1
Enter element :  34
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option :  1
Enter element :  56
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option :  1
Enter element :  45
Queue is overflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option :  3
Elements in the queue : 34 56  2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option :  2
```

Deleted element = 34 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 56 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Queue is underflow. 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Queue is empty. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 56
Successfully inserted. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Elements in the queue : 56 4
1.Enqueue 2.Dequeue 3.Display 4.Exit 4
Enter your option : 4

S.No: 19	Exp. Name: ***Write a C program to implement different Operations on Queue using Array representation***	Date:2023-06-13

ID: 224G1A0560

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

Aim:

Write a program to implement queue using **arrays**.

```
Sample Input and Output:
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 1
        Enter element : 23
        Successfully inserted.
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 1
        Enter element : 56
        Successfully inserted.
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 3
        Elements in the queue : 23 56
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 4
        Queue is not empty.
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 5
        Queue size : 2
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 2
        Deleted element = 23
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 2
        Deleted element = 56
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 4
        Queue is empty.
        1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit
        Enter your option : 6
```

Source Code:

QUsingArray.c

```
#include <conio.h>
#include <stdio.h>
#define MAX 10
int queue[MAX];
int front=-1,rear=-1;
void enqueue(int X)
{
   if(rear==MAX-1)
   {
      printf("Queue is overflowed.\n");
   }
else
{
   rear++;
```

```
   queue[rear]=X;
   printf("Successfully inserted.\n");
}
if(front==-1)
{
   front++;
}
}
void dequeue()
{
   if(front==-1)
   {
      printf("Queue is underflow.\n");
   }
   else
   {
      printf("Deleted element = %d\n",queue[front]);
      if(rear==front)
      {
         rear=front=-1;
      }
   else
   {
      front++;
   }
   }
   }
void display()
{
   if(front==-1&&rear==-1)
   {
      printf("Queue is empty.\n");
   }
else
{
   printf("Elements in the queue : ");
   for(int i=front;i<=rear;i++)
   {
      printf("%d ",queue[i]);
   }
   printf("\n");
   }
}
void size()
{
   if(front==-1&&rear==-1)
   printf("Queue size : 0\n");
   else
   printf("Queue size : %d\n",rear-front+1);
}
void isEmpty()
{
   if(front==-1&&rear==-1)
   printf("Queue is empty.\n");
   else
   printf("Queue is not empty.\n");
```


ID: 224G1A0560
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

```
}
int main()
{
   int op,X;
   while(1)
   {
      printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
      printf("Enter your option : ");
      scanf("%d",&op);
      switch(op)
      {
         case 1:
         printf("Enter element : ");
         scanf("%d",&X);
         enqueue(X);
         break;
         case 2:
         dequeue();
         break;
         case 3:
         display();
         break;
         case 4:
         isEmpty();
         break;
         case 5:
         size();
         break;
         case 6:
         exit(0);
      }
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option : 2
Queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option : 3
Queue is empty. 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Enter your option : 4
Queue is empty. 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Enter your option : 5
Queue size : 0 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1

```
Enter your option :  1
Enter element :  14
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option :  1
Enter element :  78
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option :  1
Enter element :  53
Successfully inserted. 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option :  3
Elements in the queue : 14 78 53  5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Enter your option :  5
Queue size : 3 6
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 6
Enter your option :  6
```

Test Case - 2

User Output

```
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option :  1
Enter element :  25
Successfully inserted. 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option :  2
Deleted element = 25 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option :  2
Queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option :  3
Queue is empty. 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option :  1
Enter element :  65
Successfully inserted. 3
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 3
Enter your option :  3
Elements in the queue : 65  4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Enter your option :  4
Queue is not empty. 2
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 2
Enter your option :  2
Deleted element = 65 4
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 4
Enter your option :  4
Queue is empty. 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
```

```
Enter your option :  5
Queue size : 0 1
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 1
Enter your option :  1
Enter element :  63
Successfully inserted. 5
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 5
Enter your option :  5
Queue size : 1 6
1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit 6
Enter your option :  6
```

S.No: 18	Exp. Name: ***Write a C program to implement different Operations on Stack using Linked Lists***	Date:2023-06-16

ID: 224G1A0560

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

Aim:

Write a program to implement stack using **linked lists**.

```
Sample Input and Output:
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 1
        Enter element : 33
        Successfully pushed.
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 1
        Enter element : 22
        Successfully pushed.
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 1
        Enter element : 55
        Successfully pushed.
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 1
        Enter element : 66
        Successfully pushed.
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 3
        Elements of the stack are : 66 55 22 33
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 2
        Popped value = 66
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 2
        Popped value = 55
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 3
        Elements of the stack are : 22 33
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 5
        Peek value = 22
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 4
        Stack is not empty.
        1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
        Enter your option : 6
```

Source Code:

StackUsingLList.c

```
#include <stdio.h>
#include <stdlib.h>
struct stack
{
   int data;
   struct stack *next;
```

```
};
typedef struct stack *stk;
stk top=NULL;
stk push(int x)
{
   stk temp;
   temp = (stk)malloc(sizeof(struct stack));
   if(temp==NULL)
   {
      printf("Stack is overflow.\n");
   }
else
{
   temp->data=x;
   temp->next=top;
   top=temp;
   printf("Successfully pushed.\n");
}
}
void display()
{
   stk temp=top;
   if(temp==NULL)
   {
      printf("Stack is empty.\n");
   }
   else
   {
      printf("Elements of the stack are : ");
      while(temp!=NULL)
      {
         printf("%d ",temp->data);
         temp=temp->next;
      }
      printf("\n");
   }
}
stk pop()
{
   stk temp;
   if(top==NULL)
{
   printf("Stack is underflow.\n");
}
   else
   {
      temp=top;
      top=top->next;
      printf("Popped value = %d\n",temp->data);
      free(temp);
   }
   }
   void peek()
   {
      stk temp;
      if(top==NULL)
```

```
    {
        printf("Stack is underflow.\n");
    }
    else
    {
        temp=top;
        printf("Peek value = %d\n",temp->data);
    }
}
void isEmpty()
{
    if(top==NULL)
    {
        printf("Stack is empty.\n");
    }
    else
    {
        printf("Stack is not empty.\n");
    }
}
int main()
{
    int op,x;
    while(1)
    {
        printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");
        printf("Enter your option : ");
        scanf("%d",&op);
        switch(op)
        {
            case 1:
            printf("Enter element : ");
            scanf("%d",&x);
            push(x);
            break;
            case 2:
            pop();
            break;
            case 3:
            display();
            break;
            case 4:
            isEmpty();
            break;
            case 5:
            peek();
            break;
            case 6:
            exit(0);
        }
    }
}
```


ID: 224G1A0560

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option : 1
Enter element : 33
Successfully pushed. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option : 1
Enter element : 22
Successfully pushed. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option : 1
Enter element : 55
Successfully pushed. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option : 1
Enter element : 66
Successfully pushed. 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option : 3
Elements of the stack are : 66 55 22 33 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option : 2
Popped value = 66 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option : 2
Popped value = 55 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option : 3
Elements of the stack are : 22 33 5
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 5
Enter your option : 5
Peek value = 22 4
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 4
Enter your option : 4
Stack is not empty. 6
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 6
Enter your option : 6

Test Case - 2
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option : 2
Stack is underflow. 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option : 3

```
Stack is empty. 5
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 5
Enter your option :  5
Stack is underflow. 4
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 4
Enter your option :  4
Stack is empty. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option :  1
Enter element :  23
Successfully pushed. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option :  1
Enter element :  24
Successfully pushed. 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option :  3
Elements of the stack are : 24 23  5
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 5
Enter your option :  5
Peek value = 24 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option :  2
Popped value = 24 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option :  2
Popped value = 23 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option :  2
Stack is underflow. 4
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 4
Enter your option :  4
Stack is empty. 6
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 6
Enter your option :  6
```

S.No: 17	Exp. Name: ***Write a C program to implement different Operations on Stack using Array representation***	Date:2023-06-11

ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

Aim:

Write a program to implement `stack` using **arrays**.

```
Sample Input and Output:
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 4
    Stack is empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Stack is underflow.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Stack is empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 5
    Stack is underflow.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 25
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 1
    Enter element : 26
    Successfully pushed.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 3
    Elements of the stack are : 26 25
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 2
    Popped value = 26
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 4
    Stack is not empty.
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 5
    Peek value = 25
    1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit
    Enter your option : 6
```

Source Code:

StackUsingArray.c

```
#include <stdio.h>
#include <stdlib.h>
#define STACK_MAX_SIZE 10
int arr[STACK_MAX_SIZE];
int top = -1;
void push(int element)
{
   if(top == STACK_MAX_SIZE-1)
```

```
   {
      printf("Stack is overflow.\n");
   }
   else
   {
      top = top + 1;
      arr[top] = element;
      printf("Successfully pushed.\n");
   }
}
void display()
{
   if(top < 0)
   {
      printf("Stack is empty.\n");
   }
   else
   {
      printf("Elements of the stack are : ");
      for(int i = top; i >= 0; i--)
      {
         printf("%d ",arr[i]);
      }
      printf("\n");
      }
   }
void pop()
{
   int x;
   if(top < 0)
   {
      printf("Stack is underflow.\n");
   }
else
{
   x = arr[top];
   top = top - 1;
   printf("Popped value = %d\n",x);
}
}
void peek()
{
   int x;
   if(top < 0)
   {
      printf("Stack is underflow.\n");
   }
   else
   {
      x = arr[top];
      printf("Peek value = %d\n",x);
   }
}
void isEmpty()
   {
      if(top < 0)
```

```
    {
       printf("Stack is empty.\n");
    }
    else
    {
       printf("Stack is not empty.\n");
    }
}
int main()
{
    int op,x;
    while(1)
    {
       printf("1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit\n");
       printf("Enter your option : ");
       scanf("%d",&op);
       switch(op)
       {
          case 1:
          printf("Enter element : ");
          scanf("%d",&x);
          push(x);
          break;
       case 2:
       pop();
       break;
       case 3:
       display();
       break;
       case 4:
       isEmpty();
       break;
       case 5:
       peek();
       break;
       case 6:
       exit(0);
    }
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option : 1
Enter element : 10
Successfully pushed. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option : 1
Enter element : 20

ID: 224G1A0560
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

```
Successfully pushed. 1
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 1
Enter your option :  1
Enter element :  30
Successfully pushed. 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option :  3
Elements of the stack are : 30 20 10  5
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 5
Enter your option :  5
Peek value = 30 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option :  2
Popped value = 30 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option :  2
Popped value = 20 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option :  3
Elements of the stack are : 10  5
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 5
Enter your option :  5
Peek value = 10 4
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 4
Enter your option :  4
Stack is not empty. 2
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 2
Enter your option :  2
Popped value = 10 3
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 3
Enter your option :  3
Stack is empty. 4
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 4
Enter your option :  4
Stack is empty. 6
1.Push 2.Pop 3.Display 4.Is Empty 5.Peek 6.Exit 6
Enter your option :  6
```

S.No: 16	Exp. Name: ***Implementation of Circular Queue using Dynamic Array***	Date:2023-06-12

Aim:

Write a program to implement `circular queue` using **dynamic array**.

ID: 224G1A0560

2022-2026-CSE-A

Srinivasa Ramanujan Institute of Technology

```
Sample Input and Output:
    Enter the maximum size of the circular queue : 3
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Circular queue is underflow.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Circular queue is empty.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 111
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 222
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 333
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 444
    Circular queue is overflow.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Elements in the circular queue : 111 222 333
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 111
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 1
    Enter element : 444
    Successfully inserted.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Elements in the circular queue : 222 333 444
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 222
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 333
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 2
    Deleted element = 444
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 3
    Circular queue is empty.
    1.Enqueue 2.Dequeue 3.Display 4.Exit
    Enter your option : 4
```

Source Code:

CQueueUsingDynamicArray.c

```
#include <stdio.h>
#include <stdlib.h>
int *cqueue;
int front,rear;
int maxsize;
void initcircularqueue()
{
   cqueue=(int *)malloc(maxsize * sizeof(int));
   front=-1;
   rear=-1;
}
void dequeue()
{
   if(front==-1)
   {
      printf("Circular queue is underflow.\n");
   }
   else
   {
      printf("Deleted element = %d\n",*(cqueue+front));
      if(rear==front)
      {
         rear=front=-1;
      }
      else if(front==maxsize-1)
   {
      front=0;
   }
   else
   {
      front++;
   }
}
}
void enqueue(int x)
{
   if(((rear==maxsize-1)&&(front==0))||(rear+1==front))
   {
      printf("Circular queue is overflow.\n");
   }
   else
   {
      if(rear==maxsize-1)
      {
         rear=-1;
      }
      else if(front==-1)
      {
         front=0;
      }
   rear++;
   cqueue[rear]=x;
   printf("Successfully inserted.\n");
   }
```


ID: 224G1A0560

```
}
void display()
{
   int i;
   if(front==-1&&rear==-1)
   {
      printf("Circular queue is empty.\n");
   }
   else
   {
      printf("Elements in the circular queue : ");
      if(front<=rear)
      {
         for(i=front;i<=rear;i++)
         {
            printf("%d ",*(cqueue + i));
         }
      }
      else
      {
         for(i=front;i<=maxsize-1;i++)
         {
            printf("%d ",*(cqueue + i));
         }
         for(i=0;i<=rear;i++)
      {
      printf("%d ",*(cqueue + i));
      }
         }
      printf("\n");
   }
   }
   int main()
   {
      int op,x;
      printf("Enter the maximum size of the circular queue : ");
      scanf("%d", &maxsize);
      initcircularqueue();
      while(1)
   {
      printf("1.Enqueue 2.Dequeue 3.Display 4.Exit\n");
      printf("Enter your option : ");
      scanf("%d",&op);
      switch(op)
      {
         case 1:
         printf("Enter element : ");
         scanf("%d",&x);
         enqueue(x);
         break;
         case 2:
         dequeue();
         break;
         case 3:
         display();
         break;
```


ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

```
        case 4:
        exit(0);
      }
      }
   }
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the maximum size of the circular queue : 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Circular queue is underflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Circular queue is empty. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 111
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 222
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 333
Successfully inserted. 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 444
Circular queue is overflow. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Elements in the circular queue : 111 222 333 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 111 1
1.Enqueue 2.Dequeue 3.Display 4.Exit 1
Enter your option : 1
Enter element : 444
Successfully inserted. 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Elements in the circular queue : 222 333 444 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 222 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2

Enter your option : 2
Deleted element = 333 2
1.Enqueue 2.Dequeue 3.Display 4.Exit 2
Enter your option : 2
Deleted element = 444 3
1.Enqueue 2.Dequeue 3.Display 4.Exit 3
Enter your option : 3
Circular queue is empty. 4
1.Enqueue 2.Dequeue 3.Display 4.Exit 4
Enter your option : 4

S.No: 15	Exp. Name: ***C program to which performs all operations on Circular linked list.***	Date:2023-06-11

Aim:

Write a program that uses functions to perform the following **operations on Circular linked list**
i)Creation ii)insertion iii)deletion iv) Traversal

Source Code:

AlloperationsinCLL.c

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
void insert();
void deletion();
void find();
void print();
struct node *head = NULL;
int main()
{
   int choice;
   printf("CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT\n");
   while(1)
   {
      printf("1.INSERT ");
      printf("2.DELETE ");
      printf("3.FIND ");
      printf("4.PRINT ");
      printf("5.QUIT\n");
      printf("Enter the choice: ");
      scanf("%d", &choice);
      switch(choice)
      {
         case 1:insert();break;
         case 2:deletion();break;
         case 3:find();break;
         case 4:print();break;
         case 5:exit(0);
      }
   }
}
void insert()
{
   int x,n;
   struct node *newnode,*temp = head, *prev;
   newnode = (struct node*)malloc(sizeof(struct node));
   printf("Enter the element to be inserted: ");
   scanf("%d" ,&x);
   printf("Enter the position of the element: ");
   scanf("%d" ,&n);
   newnode->data = x;
   newnode->next = NULL;
```


ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

```
    if(head == NULL)
    {
       head = newnode;
       newnode->next = newnode;
    }
    else if(n == 1)
    {
       temp = head;
       newnode->next = temp;
       while(temp->next != head)
       temp = temp->next;
       temp->next = newnode;
       head = newnode;
    }
    else
    {
       for(int i = 1;i < n-1; i++)
       {
          temp = temp->next;
       }
       newnode->next = temp->next;
       temp->next = newnode;
       }
    }
void deletion()
{
    struct node *temp = head, *prev, *temp1 = head;
    int key, count = 0;
    printf("Enter the element to be deleted: ");
    scanf("%d", &key);
    if(temp->data == key)
    {
       prev = temp -> next;
       while(temp->next != head)
       {
          temp = temp->next;
       }
       temp->next = prev;
       free(head);
       head = prev;
       printf("Element deleted\n");
    }
    else
    {
       while(temp->next != head)
       {
          if(temp->data == key)
          {
             count += 1;
             break;
          }
          prev = temp;
          temp = temp->next;
       }
       if(temp->data == key)
       {
```

```
         prev->next = temp->next;
         free(temp);
      printf("Element deleted\n");
}
else
{
   printf("Element does not exist...!\n");
}
}
}
void find()
{
   struct node *temp = head;
   int key,count = 0;
   printf("Enter the element to be searched: ");
   scanf("%d", &key);
   while(temp->next != head)
   {
      if(temp->data == key)
      {
         count = 1;
         break;
      }
      temp = temp->next;
   }
if(count == 1)
printf("Element exist...!\n");
else
{
   if(temp->data == key)
   printf("Element exist...!\n");
   else
   printf("Element does not exist...!\n");
}
}
void print()
{
   struct node *temp = head;
   printf("The list element are: ");
   while(temp->next != head)
   {
      printf("%d -> ",temp->data);
      temp = temp->next;
   }
printf("%d -> ",temp->data);
printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT 1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1

ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

```
Enter the choice:  1
Enter the element to be inserted:  12
Enter the position of the element:  1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  14
Enter the position of the element:  2
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  15
Enter the position of the element:  3
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 12 -> 14 -> 15 ->  2
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 2
Enter the choice:  2
Enter the element to be deleted:  14
Element deleted 4
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 12 -> 15 ->  3
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 3
Enter the choice:  3
Enter the element to be searched:  12
Element exist...! 5
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 5
Enter the choice:  5
```

Test Case - 2

User Output

```
CIRCULAR LINKED LIST IMPLEMENTATION OF LIST ADT 1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  54
Enter the position of the element:  1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 2
Enter the choice:  2
Enter the element to be deleted:  1
Element does not exist...! 4
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 54 ->  1
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 1
Enter the choice:  1
Enter the element to be inserted:  65
Enter the position of the element:  2
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 4
Enter the choice:  4
The list element are: 54 -> 65 ->  5
1.INSERT 2.DELETE 3.FIND 4.PRINT 5.QUIT 5
Enter the choice:  5
```

S.No: 14	Exp. Name: ***C program which performs all operations on double linked list.***	Date:2023-07-22

Aim:

Write a C program that uses functions to perform the following **operations on double linked list**
i) Creation ii) Insertion iii) Deletion iv) Traversal

Source Code:

AllOperationsDLL.c

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

struct dnode
{
   struct dnode *prev;
   int data;
   struct dnode *next;
};

struct dnode *start = NULL;

void insert(int);
void remov(int);
void display();

int main()
{
   int n, ch;
   do
   {
      printf("Operations on doubly linked list");
      printf("\n1. Insert \n2.Remove\n3. Display\n0. Exit");
      printf("\nEnter Choice 0-4? : ");
      scanf("%d", &ch);
      switch (ch)
      {
         case 1:
         printf("Enter number: ");
         scanf("%d", &n);
         insert(n);
         break;
         case 2:
         printf("Enter number to delete: ");
         scanf("%d", &n);
         remov(n);
         break;
         case 3:
         display();
         break;
      }
   }while (ch != 0);
}
```


ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

```
void insert(int num)
{
   struct dnode *nptr,  *temp = start;


       nptr = malloc(sizeof(struct dnode));
       nptr->data = num;
       nptr->next = NULL;
       nptr->prev = NULL;


           if (start == NULL)
           {
            start = nptr;
           }
           else
           {

                   while (temp->next != NULL)
                   temp = temp->next;

                   nptr->prev = temp;
                   temp->next = nptr;
           }
}


void remov(int num)
{
   struct dnode *temp = start;

   while (temp != NULL)
   {
      if (temp->data == num)
      {

                     if (temp == start)
                     {
                        start = start->next;
                        start->prev = NULL;
                     }
                     else
                     {

                                        if (temp->next == NULL)
                                        temp->prev->next = NULL;
                                        else
                                        {
                     temp->prev->next = temp->next;
                     temp->next->prev = temp->prev;
                     }
                     free(temp);
      }
      return ;
   }
   temp = temp->next;
```

```
}
printf("%d not found.\n", num);
}


void display()
{
   struct dnode *temp = start;
   while (temp != NULL)
   {
      printf("%d\t", temp->data);
      temp = temp->next;
   }
   printf("\n");
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 15
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 16
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 17
Operations on doubly linked list 1
1.Insert 1
2.Remove 1
3.Display 1
0.Exit 1
Enter Choice 0-4?: 1
Enter number: 18
Operations on doubly linked list 3
1.Insert 3
2.Remove 3
3.Display 3

0.Exit 3
Enter Choice 0-4?: 3
15 16 17 18 2
Operations on doubly linked list 2
1.Insert 2
2.Remove 2
3.Display 2
0.Exit 2
Enter Choice 0-4?: 2
Enter number to delete: 19
19 not found 3
Operations on doubly linked list 3
1.Insert 3
2.Remove 3
3.Display 3
0.Exit 3
Enter Choice 0-4?: 3
15 16 17 18 2
Operations on doubly linked list 2
1.Insert 2
2.Remove 2
3.Display 2
0.Exit 2
Enter Choice 0-4?: 2
Enter number to delete: 16
Operations on doubly linked list 0
1.Insert 0
2.Remove 0
3.Display 0
0.Exit 0
Enter Choice 0-4?: 0

S.No: 13	Exp. Name: ***C program to performs all operations on singly linked list***	Date:2023-07-22

Aim:

Write a program that uses functions to perform the following **operations on singly linked list**

i) Creation

ii) Insertion

iii) Deletion

iv) Traversal

Source Code:

singlelinkedlistalloperations.c

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
struct node {
   int value;
   struct node *next;
};
void insert();
void display();
void delete();
int count();
typedef struct node DATA_NODE;
DATA_NODE *head_node, *first_node, *temp_node = 0, *prev_node, next_node;
int data;
int main() {
   int option = 0;
   printf("Singly Linked List Example - All Operations\n");
   while (option < 5) {
      printf("Options\n");
      printf("1 : Insert elements into the linked list\n");
      printf("2 : Delete elements from the linked list\n");
      printf("3 : Display the elements in the linked list\n");
      printf("4 : Count the elements in the linked list\n");
      printf("5 : Exit()\n");
      printf("Enter your option : ");
      scanf("%d", &option);
      switch (option) {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 4:
         count();
         break;
         case 5:
         exit(0);
         break;
```

```
         default:
         printf("Enter options from 1 to 5\n");
         break;
      }
   }
   return 0;
}
void insert() {
   printf("Enter elements for inserting into linked list : ");
   scanf("%d", &data);
   temp_node = (DATA_NODE *) malloc(sizeof (DATA_NODE));
   temp_node->value = data;
   if (first_node == 0) {
      first_node = temp_node;
   } else {
      head_node->next = temp_node;
   }
   temp_node->next = 0;
   head_node = temp_node;
   fflush(stdin);
}
void delete() {
   int countvalue, pos, i = 0;
   temp_node = first_node;
   printf("Enter position of the element for deleteing the element : ");
   scanf("%d", &pos);
   if (pos > 0 && pos <= countvalue) {
      if (pos == 1) {
         temp_node = temp_node -> next;
         first_node = temp_node;
         printf("Deleted successfully\n");
      } else {
         while (temp_node != 0) {
            if (i == (pos - 1)) {
               prev_node->next = temp_node->next;
               if (i == (countvalue - 1)) {
                  head_node = prev_node;
               }
               printf("Deleted successfully\n");
               break;
            } else {
               i++;
               prev_node = temp_node;
               temp_node = temp_node -> next;
            }
         }
      }
   } else
   printf("Invalid position\n");
}
void display() {
   int count = 0;
   temp_node = first_node;
   printf("The elements in the linked list are : ");
   while (temp_node != 0) {
      printf("%d ", temp_node->value);
```

```
      temp_node = temp_node -> next;
   }
   printf("\n");
}
int count() {
   int count = 0;
   temp_node = first_node;
   while (temp_node != 0) {
      count++;
      temp_node = temp_node -> next;
   }
   printf("No of elements in the linked list are : %d\n", count);
   return count;
}
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Singly Linked List Example - All Operations 1
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 111
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 222
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 333
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 444

Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option : 3
The elements in the linked list are : 111 222 333 444 2
Options 2
1 : Insert elements into the linked list 2
2 : Delete elements from the linked list 2
3 : Display the elements in the linked list 2
4 : Count the elements in the linked list 2
5 : Exit() 2
Enter your option : 2
Enter position of the element for deleteing the element : 2
Deleted successfully 3
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option : 3
The elements in the linked list are : 111 333 444 4
Options 4
1 : Insert elements into the linked list 4
2 : Delete elements from the linked list 4
3 : Display the elements in the linked list 4
4 : Count the elements in the linked list 4
5 : Exit() 4
Enter your option : 4
No of elements in the linked list are : 3 5
Options 5
1 : Insert elements into the linked list 5
2 : Delete elements from the linked list 5
3 : Display the elements in the linked list 5
4 : Count the elements in the linked list 5
5 : Exit() 5
Enter your option : 5

Test Case - 2
User Output
Singly Linked List Example - All Operations 1
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option : 1
Enter elements for inserting into linked list : 001

```
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  010
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  100
Options 1
1 : Insert elements into the linked list 1
2 : Delete elements from the linked list 1
3 : Display the elements in the linked list 1
4 : Count the elements in the linked list 1
5 : Exit() 1
Enter your option :  1
Enter elements for inserting into linked list :  101
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option :  3
The elements in the linked list are : 1 10 100 101  2
Options 2
1 : Insert elements into the linked list 2
2 : Delete elements from the linked list 2
3 : Display the elements in the linked list 2
4 : Count the elements in the linked list 2
5 : Exit() 2
Enter your option :  2
Enter position of the element for deleteing the element :  3
Deleted successfully 3
Options 3
1 : Insert elements into the linked list 3
2 : Delete elements from the linked list 3
3 : Display the elements in the linked list 3
4 : Count the elements in the linked list 3
5 : Exit() 3
Enter your option :  3
The elements in the linked list are : 1 10 101  4
Options 4
1 : Insert elements into the linked list 4
2 : Delete elements from the linked list 4
3 : Display the elements in the linked list 4
```

4 : Count the elements in the linked list 4
5 : Exit() 4
Enter your option : 4
No of elements in the linked list are : 3 5
Options 5
1 : Insert elements into the linked list 5
2 : Delete elements from the linked list 5
3 : Display the elements in the linked list 5
4 : Count the elements in the linked list 5
5 : Exit() 5
Enter your option : 5

S.No: 12	Exp. Name: ***Write a C program to sort given elements using Radix sort***	Date:2023-05-07

Aim:

Write a program to `sort` (`ascending order`) the given elements using `radix sort` technique.

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the **input** as:

```
Enter array size : 5
```

Next, the program should print the following message on the console as:

```
Enter 5 elements :
```

if the user gives the **input** as:

```
Enter 5 elements : 34 67 12 45 22
```

then the program should **print** the result as:

```
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67
```

Note: Do use the **printf()** function with a **newline** character (`\n`).

Source Code:

RadixSortMain2.c

```
#include <stdio.h>
#include <conio.h>
int main() {
   int size;
   int *arr, i;
   printf("Enter array size : ");
   scanf("%d",&size);
   arr = (int*) malloc(size * sizeof(int));
   printf("Enter %d elements : ",size);
   for (i = 0; i < size; i++) {
      scanf("%d", &arr[i]);
   }
   printf("Before sorting the elements are : ");
   printArray(arr,size);
   RadixSort(arr,size);
   printf("After sorting the elements are : ");
   printArray(arr,size);
   return 0;
}
int largest(int a[], int n) {
   int i, k = a[0];
   for(i=1;i<n;i++) {
   if(a[i]>k) {
      k = a[i];
   }
```

```
}
return k;
}
void printArray(int a[], int n) {
   int i;
   for(i=0;i<n;i++) {
   printf("%d ",a[i]);
}
printf("\n");
}
void RadixSort(int a[], int n) {
   int bucket[10][10],bucket_count[10],i,j,k,rem,NOP=0,divi=1,large,pass;
   large=largest(a,n);
   while(large>0) {
      NOP++;
      large/=10;
   }
   for(pass=0;pass<NOP;pass++) {
      for(i=0;i<=10;i++) {
         bucket_count[i] = 0;
      }
   for(i=0;i<n;i++) {
      rem = (a[i]/divi)%10;
      bucket[rem][bucket_count[rem]] = a[i];
      bucket_count[rem]++;
   }
   i=0;
   for(k=0;k<10;k++) {
   for(j=0;j<bucket_count[k];j++) {
      a[i] = bucket[k][j];
      i++;
   }
   }
   divi*=10;
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size : 5
Enter 5 elements : 23
43
54
12
65
Before sorting the elements are : 23 43 54 12 65
After sorting the elements are : 12 23 43 54 65

ID: 224G1A0560
2022-2026-CSE-A
Srinivasa Ramanujan Institute of Technology

Test Case - 2
User Output
Enter array size : 7
Enter 7 elements : 23
54
136
85
24
65
76
Before sorting the elements are : 23 54 136 85 24 65 76
After sorting the elements are : 23 24 54 65 76 85 136

ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

S.No: 11	Exp. Name: ***Write a C program to Sort given elements using Merge sort***	Date:2023-05-10

ID: 224G1A0560

Aim:

Write a program to `sort` (`Ascending order`) the given elements using `merge sort` technique.

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the **input** as:

```
Enter array size : 5
```

Next, the program should print the following message on the console as:

```
Enter 5 elements :
```

if the user gives the **input** as:

```
Enter 5 elements : 34 67 12 45 22
```

then the program should **print** the result as:

```
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67
```

Note: Do use the **printf()** function with a **newline** character (`\n`).

Source Code:

MergeSortMain.c

```
#include <stdio.h>
void main() {
   int arr[15], i, n;
   printf("Enter array size : ");
   scanf("%d", &n);
   printf("Enter %d elements : ",n);
   for (i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }
   printf("Before sorting the elements are : ");
   display(arr, n);
   splitAndMerge(arr, 0, n - 1);
   printf("After sorting the elements are : ");
   display(arr, n);
   }
void display(int arr[15], int n) {
   int i;
   for(i = 0; i < n; i++)
   printf("%d ",arr[i]);
   printf("\n");
}
void merge(int arr[15], int low, int mid, int high) {
   int i = low, h = low,j = mid + 1, k, temp[15];
   while (h <= mid && j<= high)
   {
```

```
      if(arr[h] <= arr[j])
      {
         temp[i] = arr[h];
         h++;
      }
      else
      {
         temp[i] = arr[j];
         j++;
      }
      i++;
   }
   if (h > mid)
   {
      for (k = j; k <= high; k++)
      {
         temp[i] = arr[k];
         i++;
      }
   }
   else
   {
      for (k = h; k <= mid; k++)
      {
         temp[i] = arr[k];
         i++;
      }
   }
   for (k = low; k <= high; k++){
      arr[k] = temp[k];
   }
}
void splitAndMerge(int arr[15], int low, int high) {
   if(low < high){
      int mid = (low + high) / 2;
      splitAndMerge(arr, low, mid);
      splitAndMerge(arr, mid+1, high);
      merge(arr, low, mid, high);
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size : 5

ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

Enter 5 elements : 34 67 12 45 22
Before sorting the elements are : 34 67 12 45 22
After sorting the elements are : 12 22 34 45 67

Test Case - 2
User Output
Enter array size : 8
Enter 8 elements : 77 55 22 44 99 33 11 66
Before sorting the elements are : 77 55 22 44 99 33 11 66
After sorting the elements are : 11 22 33 44 55 66 77 99

Test Case - 3
User Output
Enter array size : 5
Enter 5 elements : -32 -45 -67 -46 -14
Before sorting the elements are : -32 -45 -67 -46 -14
After sorting the elements are : -67 -46 -45 -32 -14

S.No: 10	Exp. Name: ***Write a C program to sort the given elements using Heap sort***	Date:2023-05-20

Aim:

Write a program to sort (ascending order) the given elements using heap sort technique.

Note: Do use the printf() function with a newline character (\n).

Source Code:

HeapSortMain.c

```
#include <stdio.h>
void display();
void heapsort();
void main()
{
   int arr[15],i,n;
   printf("Enter array size : ");
   scanf("%d", &n);
   printf("Enter %d elements : ", n);
   for (i=0;i<n;i++)
   {
       scanf("%d",&arr[i]);
}
printf("Before sorting the elements are : ");
display(arr,n);
heapsort(arr,n);
printf("After sorting the elements are : ");
display(arr,n);
}
void display(int arr[15], int n)
{
   int i;
   for(i=0;i<n;i++)
   printf("%d ", arr[i]);
   printf("\n");
}
void heapify(int arr[], int n, int i)
{
   int largest =i;
   int l=2*i+1;
   int r=2*i+2;
   int temp;
   if (l<n&&arr[l]>arr[largest])
   largest=l;
   if (r<n&&arr[r]>arr[largest])
   largest=r;
   if (largest!=i)
   {
    temp=arr[i];
      arr[i]=arr[largest];
      arr[largest]=temp;
      heapify(arr,n,largest);
   }
}
```

```
    void heapsort(int arr[],int n)
    {
       int i,temp;
    for(i=n/2-1;i>=0;i--)
    {
       heapify(arr,n,i);
    }
    for(i=n-1;i>=0;i--)
    {
     temp=arr[0];
       arr[0]=arr[i];
       arr[i]=temp;
       heapify(arr,i,0);
    }
 }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter array size : 5
Enter 5 elements : 23 54 22 44 12
Before sorting the elements are : 23 54 22 44 12
After sorting the elements are : 12 22 23 44 54

Test Case - 2
User Output
Enter array size : 6
Enter 6 elements : 12 65 23 98 35 98
Before sorting the elements are : 12 65 23 98 35 98
After sorting the elements are : 12 23 35 65 98 98

ID: 224G1A0560

Srinivasa Ramanujan Institute of Technology

Test Case - 3
User Output
Enter array size : 4
Enter 4 elements : -23 -45 -12 -36
Before sorting the elements are : -23 -45 -12 -36
After sorting the elements are : -45 -36 -23 -12

Test Case - 4
User Output
Enter array size : 6
Enter 6 elements : 1 -3 8 -4 -2 5
Before sorting the elements are : 1 -3 8 -4 -2 5
After sorting the elements are : -4 -3 -2 1 5 8