3b) Circular Queue

```
#include <stdio.h>
#include <stdlib.h>
# define Size 50
int Q[Size];
int rear = -1;
int front = -1;
int IsFull()
{
 if(front == (rear+1) % Size)
 {
  return 0;
 }
 else
  {
   return -1;
  }
}
int IsEmpty()
{
 if(front == -1 && rear == -1)
  {
   return 0;
  }
  else
   {
    return -1;
   }
 }
 void Enqueue(int x)
 {
    int item;
    if(IsFull() == 0)
    {
      printf("Queue overflow \n");
      return;
    }
   else
   {
    if(IsEmpty() == 0)
     {
       front = 0;
       rear = 0;
     }
    else
    {
     rear = (rear+1) % Size;
    }
    Q[rear] = x;
 }
```

```
int Dequeue()
{
  int x;
  if(IsEmpty()==0)
  {
    printf("Queue underflow \n");
  }
  else
  {
    if(front == rear)
    {
      x=Q[front];
      front=-1;
      rear=-1;
    }
    else
    {
      x=Q[front];
      front=(front+1)%Size;
    }
    return x;
  }
}
void Display()
{
  int i;
  if(IsEmpty()==0)
  {
    printf("Queue is empty\n");
  }
  else
  {
    printf("Queue elements:\n");
    for(i=front; i!=rear; i=(i+1)%Size)
    {
      printf("%d\n", Q[i]);
    }
    printf("%d\n", Q[i]);
  }
}
void main()
{
  int choice, x, b;
  while(1)
  {
```

```
printf("1. Enqueue, 2. Dequeue, 3. Display, 4. Exit \n");
printf("Enter your choice: \n");
scanf("%d", &choice);
switch(choice)
{
  case 1:
    printf("Enter the number to be inserted into the queue\n");
    scanf("%d", &x);
    Enqueue(x);
    break;
  case 2:
    b = Dequeue();
    printf("%d was removed from the queue \n", b);
    break;
  case 3:
    Display();
    break;
  case 4:
    exit(1);
  default:
    printf("Invalid input \n");
}
}
```

output:

1. Enqueue, 2. Dequeue, 3. Display, 4. Exit
Enter your choice: 1
Enter the number to be inserted into the queue: 23

1. Enqueue, 2. Dequeue, 3. Display, 4. Exit
enter your choice:
1
enter the number to be inserted into the queue
13

1. Enqueue, 2. Dequeue, 3. Display, 4. Exit
enter your choice:
2
23 was removed from the queue

1. Enqueue, 2. Dequeue, 3. Display, 4. Exit
enter your choice:
3
queue elements:
13

Single Linked list

```
#include <stdio.h>
#include<stdlib.h>
  struct node
   {
     int data;
     struct node * next;
   };
   void display();
   void insert_begin();
   void insert_end();
   Void insert_pos();
   struct node *head=NULL;
   void display()
   {
     printf("Elements are : \n");
     struct node * ptr;
      if(head==NULL)
      {
        printf("List is empty \n");
        return;
      }
     else
     {
       ptr=head;
       while(ptr!=NULL)
       {
         printf("%d \n", ptr->data);
         ptr=ptr->next;
       }
     }
   }
void insert_begin()
{
  struct node * temp;
  temp=(struct node*)malloc(sizeof(struct node));
  printf("Enter the value to be inserted \n");
  scanf("%d", &temp->data);
   temp->next=NULL;
   if(head==NULL)
     head=temp;
   else
   {
     temp->next=head;
  }  head=temp;
```

```
void insert_end()
{
 struct node *temp, *ptr;
 temp = (struct node *) malloc (sizeof(struct node));
 printf("Enter the value to be inserted \n");
 scanf("%d", &temp->data);
 temp->next = NULL;
 if(head == NULL)
 {
  head = temp;
 }
 else
 {
  ptr = head;
  while (ptr->next != NULL)
  {
   ptr = ptr->next;
  }
  ptr->next = temp;
 }
}
void insert_posn()
{
 int pos, i;
 struct node * temp, * ptr;
 printf("Enter the position");
 scanf("%d", &pos);
 temp = (struct node *) malloc (sizeof(struct node));
 printf("enter the value to be inserted \n");
 scanf("%d", &temp->data);
 temp->next = NULL;
 if(pos == 0)
 {
  temp->next = head;
  head = temp;
 }
 else
 {
  for(i=0, ptr=head; i<pos-1; i++)
  {
   ptr = ptr->next;
  }
 temp->next = ptr->next;
 ptr->next = temp;
}}
```

```
void main()
{
 int choice;
 while(1)
 printf("\n1. to insert at the beginning\n
          2. to insert at the end \n
          3. to insert at the position \n
          4. to display \n 5. exit \n");
    printf("Enter your choice: \n");
     scanf("%d", &choice);
     switch(choice)
     {
      case 1:
          insertbegin();
          break
       case 2:
          insert_end();
          break;
        case3:
          insert_pos();
          break;
       case4:
          display();
          break;
       case5:
          exit(0);
          break;
       default:
          printf("Invalid choice \n");
          break;
        }
     }
```

11/1/24

o/p

1. to insert at the beginning
2. to insert at the end
3. to insert at the position
4. to display
5. exit

enter your choice:
1
enter the value to be inserted
12
1. to insert at the beginning
2. to insert at the end
3. to insert at the position
4. to display
5. exit
enter your choice
1
enter the value to be inserted
13
1. to insert at the beginning
2. to insert at the end
3. to insert at the position
4. to ~~insert at~~ display
5. exit
enter your choice
4
elements are:
12
13
1. to insert at the beginning
2. to insert at the end
3. to insert at the position
4. to display
5. exit
enter your choice:
2
enter the value to be inserted
80
1. to insert at the beginning
2. to insert at the end
3. to insert at the position
4. to display
5. exit
enter your choice:
3
enter the position 1
enter the value to be inserted
999
elements are:
12
13
80

12
999
13