WAP to Implement Singly Linked List with following operations:
a) Create a linked list.
b) Insertion of a node at first position, at any position and at end of list.
c) Display the contents of the linked list.

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

struct Node
{

    int data;
    struct Node* next;

};
struct Node* createNode(int value){
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if(newNode == NULL){
       printf("Memory allocation failed\n");
        exit(1);
   }

    newNode->data = value;
    newNode->next = NULL;
    return newNode;

}



struct Node* insertAtBeginning(struct Node* head, int value){
    struct Node* newNode = createNode(value);
    newNode->next = head;
    return newNode;

}

struct Node* insertAtAnyPos(struct Node* head, int value, int pos){
    struct Node* newNode = createNode(value);
    struct Node* temp = head;
    if(pos == 1){
```

```
        newNode->next = head;
        return newNode;
    }
    for(int i = 1; i < pos - 1; i++){
        temp = temp->next;
    }
    newNode->next = temp->next;
    temp->next = newNode;
    return head;

}

struct Node* insertAtEnd(struct Node* head, int value){
    struct Node* newNode = createNode(value);
    if(head == NULL){
        return newNode;
   }
    struct Node* temp = head;
    while(temp->next != NULL){
        temp = temp->next;
    }
    temp->next = newNode;
    return head;

}

void displayList(struct Node* head){
    struct Node* temp = head;
    while(temp != NULL){
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main(){
    struct Node* head = NULL;
    int choice, value, pos;
    while(1){
```

```
printf("1. Insert at end\n");
printf("2. Insert at beginning\n");
printf("3. Insert at any position\n");
printf("4. Display List\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice){

   case 1:

      printf("Enter the value to be inserted: ");
      scanf("%d", &value);
      head = insertAtEnd(head, value);
      break;

   case 2:
      printf("Enter the value to be inserted: ");
      scanf("%d", &value);
      head = insertAtBeginning(head, value);
      break;
   case 3:
      printf("Enter the value to be inserted: ");
      scanf("%d", &value);
      printf("Enter the position at which to insert: ");
      scanf("%d", &pos);
      head = insertAtAnyPos(head, value, pos);
      break;

   case 4:
      displayList(head);
      break;

   case 5:
      exit(0);
   default:
      printf("Invalid choice. Please try again.\n");

}
```

```
    }

    return 0;

}
```

```
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 1
Enter the value to be inserted: 23
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 4
14 -> 13 -> 12 -> 23 -> NULL
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 3
Enter the value to be inserted: 12
Enter the position at which to insert: 2
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 4
14 -> 12 -> 13 -> 12 -> 23 -> NULL
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice:
```

```
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 2
Enter the value to be inserted: 12
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 2
Enter the value to be inserted: 13
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 2
Enter the value to be inserted: 14
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
Enter your choice: 4
14 -> 13 -> 12 -> NULL
1. Insert at end
2. Insert at beginning
3. Insert at any position
4. Display List
5. Exit
```

WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display. The program should print appropriate messages for queue empty and queue overflow conditions.

```
#include<stdio.h>

#include<stdlib.h>

#define size 3

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:");
        scanf("%d", &choice);
        switch(choice)
        {
        case 1:
            printf("Enter the number to be inserted \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 choice \n");

        }
    }
}
```

```
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:1
Enter the number to be inserted
12
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:1
Enter the number to be inserted
13
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:1
Enter the number to be inserted
14
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:3
Queue elements:
12
13
14
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:1
Enter the number to be inserted
15
Queue overflow
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:2
12 was removed from the queue
1.Enqueue, 2.Dequeue, 3.Display, 4.exit
Enter your choice:3
Queue elements:
13
14
```