[Name:Valaki Jaymin D]
[RollNo: MA075]
[Date of Experiment: 29/03/2023]

Practical-7

Implementation of Circular and Doubly Linked List

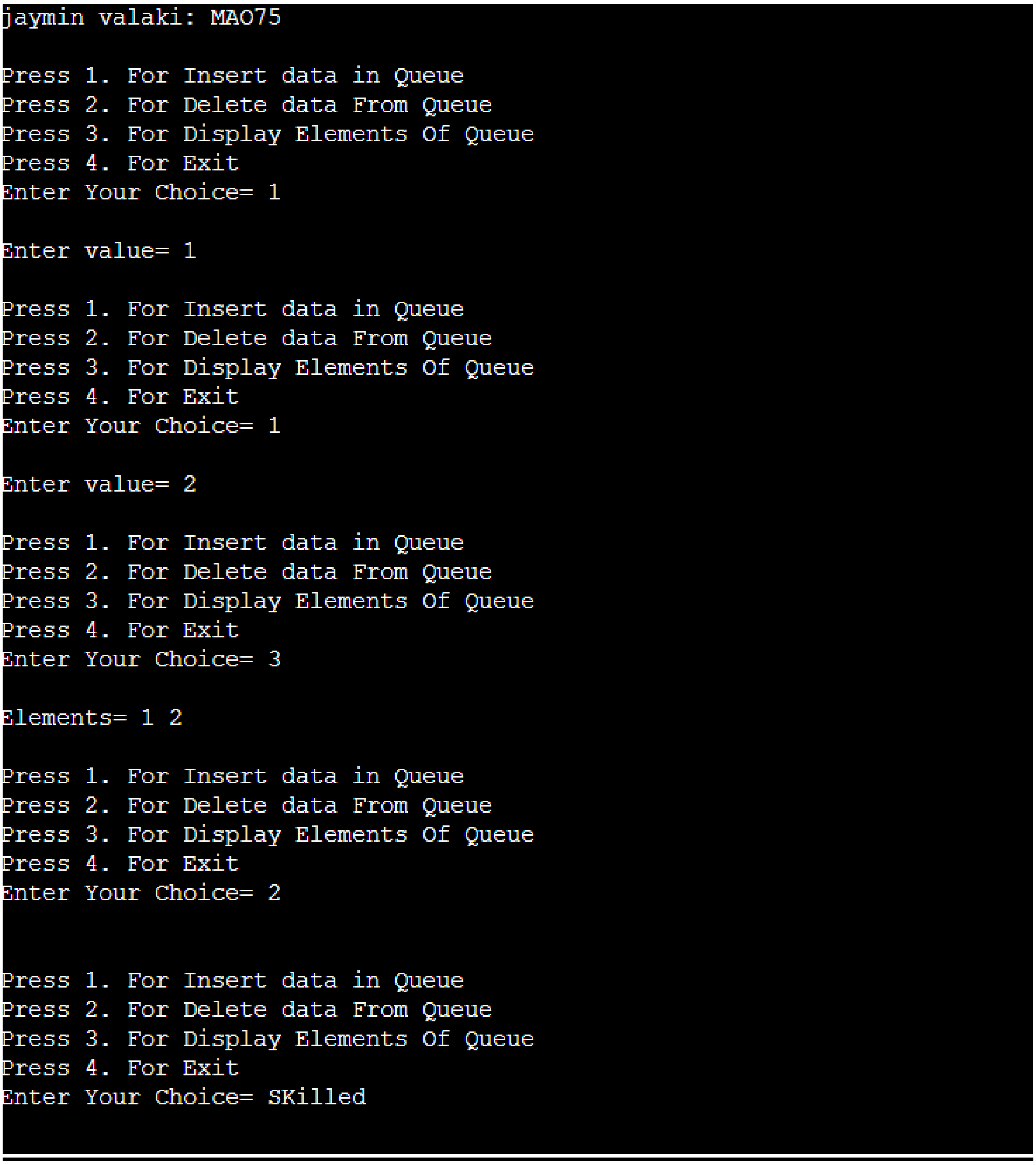
1. Write a program to implement Enqueue and Dequeue operations of circular queue using circular link list.

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int data;
	struct node* next;
};
struct node *front=NULL;
struct node *rear=NULL;
void enqueue(int data);
void dequeue();
void display();
int main()
{
	int choice,val;
	while(1)
	{
		printf("\nPress 1. For Insert data in Queue");
		printf("\nPress 2. For Delete data From Queue");
		printf("\nPress 3. For Display Elements Of Queue");
		printf("\nPress 4. For Exit");
		printf("\nEnter Your Choice= ");
		scanf("%d",&choice);
		printf("\n");
		switch(choice)
		{
			case 1: printf("Enter value= ");
			scanf("%d",&val);
			enqueue(val);
			break;
			case 2: dequeue();
			break;
			case 3: display();
			break;
			case 4: exit(0);
			default: printf("Invalid choice");
			break;
	}
}
	return 0;
}
void enqueue(int value)
{
	struct node* temp;
	temp=(struct node*)malloc(sizeof(struct node));
	temp->data=value;
	temp->next=NULL;
	if((front==NULL)&&(rear==NULL))
	{
		front=rear=temp;
		rear->next=front;
	}
	else
	{
		rear->next=temp;
		rear=temp;
		temp->next=front;
	}
}
void dequeue()
{
	struct node* temp;
	temp=front;
	if((front==NULL)&&(rear==NULL))
	{
		printf("Queue is Empty\n");
	}
	else if(front==rear)
	{
		front=rear=NULL;
		free(temp);
	}
	else
	{
		front=front->next;
		rear->next=front;
		free(temp);
	}
}
void display()
{
	struct node* temp;
	temp=front;
	if((front==NULL)&&(rear==NULL))
	{
		printf("Queue is Empty\n");
	}
	else
	{
		printf("Elements= ");
	do
	{
		printf("%d ",temp->data);
```

```
                        temp=temp->next;
            }while(temp!=front);
            printf("\n");
            }
}
```

Output:-



```
jaymin valaki: MA075

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 1

Enter value= 1

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 1

Enter value= 2

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 3

Elements= 1 2

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 2


Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= SKilled
```

[Name:Valaki Jaymin D]
[RollNo: MA075]
[Date of Experiment: 29/03/2023]

2.) Write a program for all operations of circular singly linked list.
a. Inserting Node – as First Node, at specific location, as Last Node
b. Deleting Node – at First, at Last, specific node
c. Display List

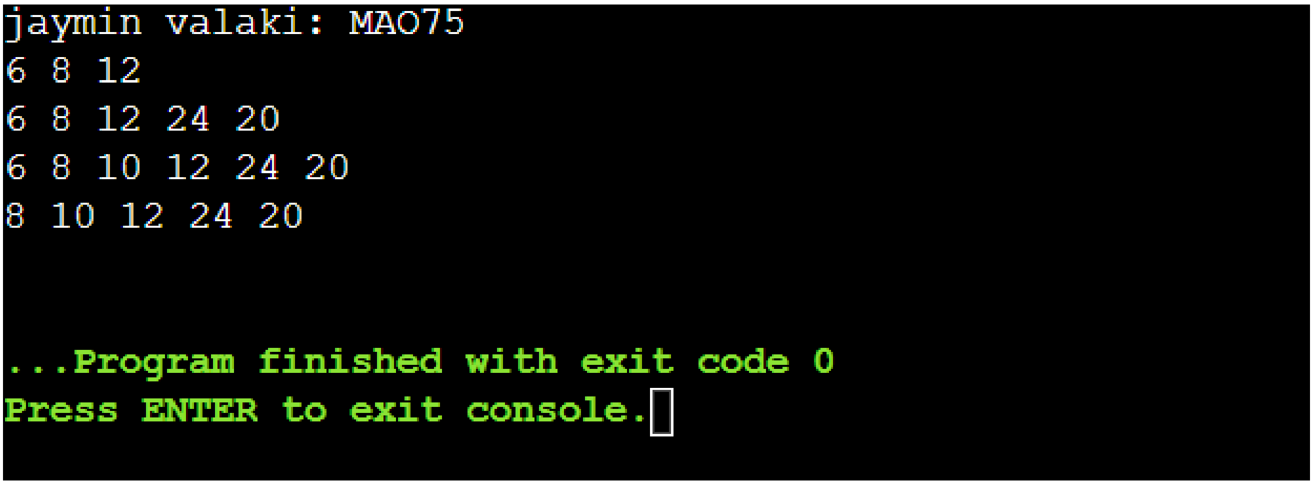
```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node *next;
};

void insertAtBeginning(struct Node **last, int data) {
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = data;
   if (*last == NULL) {
      *last = newNode;
      newNode->next = *last;
   } else {
      newNode->next = (*last)->next;
      (*last)->next = newNode;
   }
}

void insertAtEnd(struct Node **last, int data) {
   struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
   newNode->data = data;
   if (*last == NULL) {
      *last = newNode;
      newNode->next = *last;
   } else {
      newNode->next = (*last)->next;
      (*last)->next = newNode;
      *last = newNode;
   }
}

void insertAfter(struct Node **last, int key, int data) {
   struct Node *newNode, *p;
   p = (*last)->next;
   do {
      if (p->data == key) {
         newNode = (struct Node *)malloc(sizeof(struct Node));
```

```
            newNode->data = data;
            newNode->next = p->next;
            p->next = newNode;
            if (p == *last)
                *last = newNode;
            return;
        }
        p = p->next;
    } while (p != (*last)->next);
}

void deleteFirst(struct Node **last) {
    if (*last == NULL)
        return;

    struct Node *temp = (*last)->next;

    if ((*last)->next == *last) {
        free(temp);
        *last = NULL;
        return;
    }

    (*last)->next = temp->next;
    free(temp);
}

void deleteLast(struct Node **last) {
    if (*last == NULL)
        return;

    struct Node *temp, *p;

    if ((*last)->next == *last) {
        temp = *last;
        free(temp);
        *last = NULL;
        return;
    }

    p = (*last)->next;

    while (p->next != *last)
```

```
        p = p->next;

    temp = p->next;
    p->next = temp->next;
    free(temp);
}
void deleteNode(struct Node **last, int key) {
    if (*last == NULL)
        return;

    struct Node *temp, *p;

    if ((*last)->data == key) {
        temp = *last;

        if ((*last)->next == *last) {
            free(temp);
            *last = NULL;
            return;
        }

        p = (*last)->next;

        while (p->next != *last)
            p = p->next;

        p->next = temp->next;
        free(temp);
        return;
    }

    p = (*last)->next;

    while (p->next != *last) {
        if (p->next->data == key) {
            temp = p->next;
            p->next = temp->next;
            free(temp);
            return;
        }
        p = p->next;
    }
}
```

```
void display(struct Node *last) {
    struct Node *p;

    if (last == NULL) {
        printf("\nList is empty\n");
        return;
    }

    p = last->next;

    do {
        printf("%d ", p->data);
        p = p->next;
    } while (p != last->next);

    printf("\n");
}

int main() {
    struct Node *last = NULL;

    insertAtBeginning(&last, 12);
    insertAtBeginning(&last, 8);
    insertAtBeginning(&last, 6);

    display(last);

    insertAtEnd(&last, 24);
    insertAtEnd(&last, 20);

    display(last);

    insertAfter(&last, 8, 10);

    display(last);

    deleteFirst(&last);

    display(last);

    deleteLast(&last);
    display(last);

    deleteNode(&last, 10);

    display(last);

    return 0;
}
```

OUTPUT:



```
jaymin valaki: MA075
6 8 12
6 8 12 24 20
6 8 10 12 24 20
8 10 12 24 20


...Program finished with exit code 0
Press ENTER to exit console.
```

3)Write program for all operations of doubly linked list.

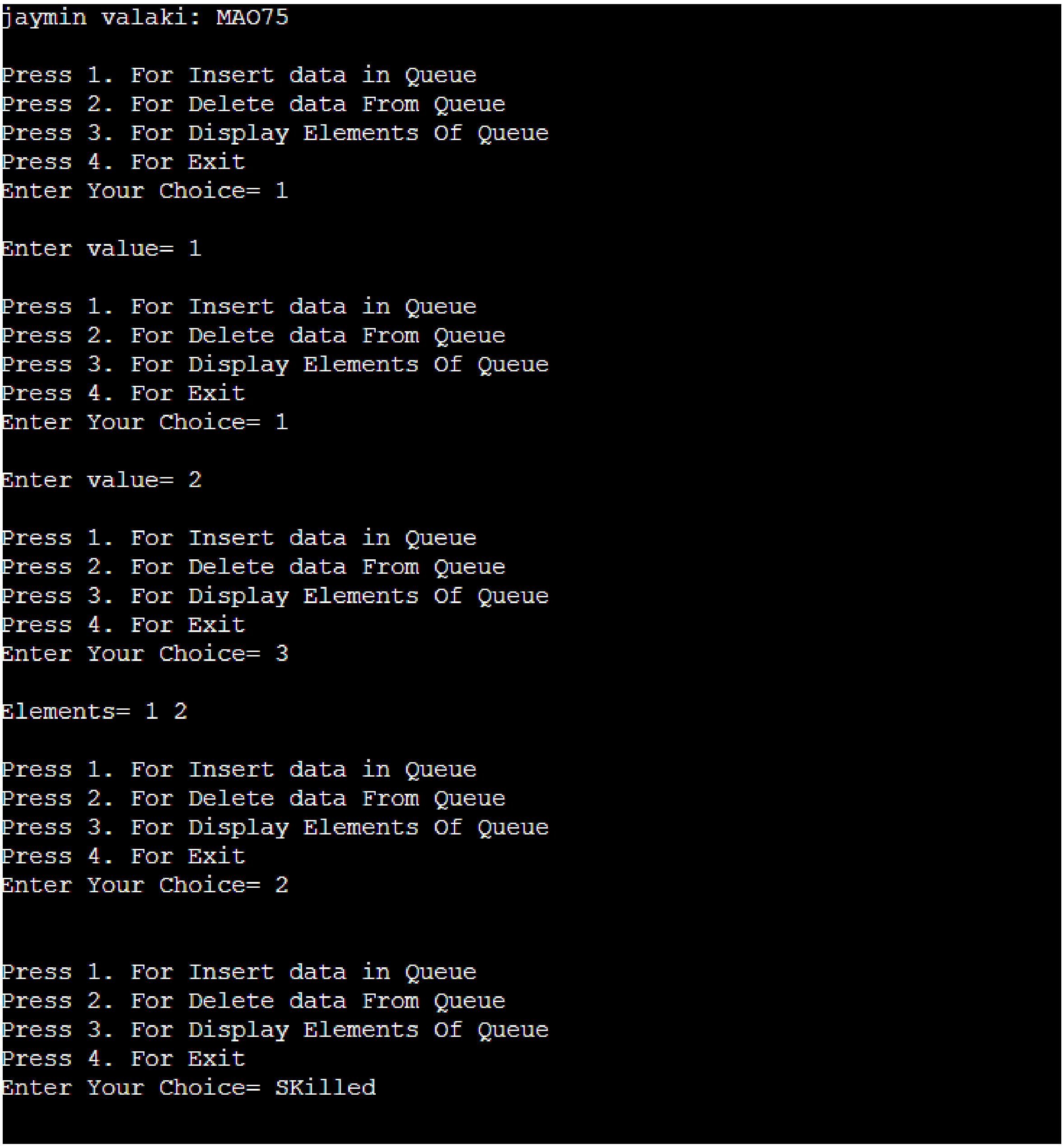
- Inserting Node – as First Node, at desired location, as Last Node
- Deleting Node – at First, at Last, Specific Node
- Display List

.

Source Code:-

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int data;
        struct node *prev, *next;
};
struct node* head = NULL;
struct node* tail = NULL;
void insertFront()
{
        int val;
        struct node* temp;
        temp=(struct node*)malloc(sizeof(struct node));
        printf(" Enter value= ");
        scanf("%d",&val);
        temp->data=val;
        temp->prev=NULL;
        temp->next=head;
        head=temp;
}
void insertEnd()
{
        int val;
        struct node *temp, *trav;
        temp=(struct node*)malloc(sizeof(struct node));
        temp->prev=NULL;
        temp->next=NULL;
        printf(" Enter value= ");
        scanf("%d",&val);
        temp->data=val;
        temp->next=NULL;
        trav=head;
        if(head==NULL)
        {
            head=temp;
        }
        else
        {
            while(trav->next!=NULL)
            {
                trav = trav->next;
            }
            temp->prev=trav;
            trav->next=temp;
        }
}
void insertPosition()
{
        int val,pos,i=1;
        struct node *temp, *newnode;
        {
        newnode=malloc(sizeof(struct node));
        newnode->next=NULL;
        newnode->prev=NULL;
        printf("Enter position= ");
        scanf("%d",&pos);
        printf("Enter value= ");
        scanf("%d",&val);
        newnode->data=val;
        temp=head;
        if(head==NULL)
        {
            head=newnode;
            newnode->prev=NULL;
            newnode->next=NULL;
        }
        else if(pos==1)
        {
            newnode->next=head;
            newnode->next->prev=newnode;
            newnode->prev=NULL;
        head=newnode;
        }
        else
        {
            while(i<pos-1)
            {
                temp = temp->next;
                i++;
            }
            newnode->next=temp->next;
            newnode->prev=temp;
            temp->next=newnode;
            temp->next->prev=newnode;
        }
}
void deleteFirst()
{
        struct node* temp;
        if(head==NULL)
        {
            printf("List is empty\n");
        }
        else
        {
            temp=head;
            head=head->next;
            if(head!=NULL)
            {
                head->prev=NULL;
            }
            free(temp);
        }
}
void deleteEnd()
{
        struct node* temp;
        if(head==NULL)
        {
            printf("List is empty\n");
        }
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        if(head->next==NULL)
        {
            head=NULL;
        }
        else
        {
            temp->prev->next=NULL;
            free(temp);
        }
}
void deletePosition()
{
        int pos,i=1;
        struct node *temp, *position;
        temp=head;
        if(head==NULL)
        {
            printf("List is empty\n");
        }
        else
        {
            printf("Enter position= ");
            scanf("%d",&pos);
            if(pos==1)
            {
                position=head;
                head=head->next;
                if(head!=NULL)
                {
                    head->prev=NULL;
                }
                free(position);
            return;
            }
            while(i<pos-1)
            {
                temp=temp->next;
                i++;
            }
            position=temp->next;
            if(position->next!=NULL)
            {
                position->next->prev=temp;
            }
            temp->next=position->next;
            free(position);
        }
}
void display()
{
        struct node* temp;
        if(head==NULL)
        {
            printf("List is empty\n");
        }
        else
        {
            printf("ELement= ");
            temp=head;
            while(temp!=NULL)
            {
                printf("%d ",temp->data);
                temp=temp->next;
            }
            printf("\n");
        }
}
int main()
{
        int choice;
        while (1)
        {
            printf("\n------------Doubly Linked List-----------\n");
            printf("\nPress 1. For Insert at first");
            printf("\nPress 2. For Insert at end ");
            printf("\nPress 3. For Insert at desired location");
            printf("\nPress 4. For Delete at first");
            printf("\nPress 5. For Delete at end");
            printf("\nPress 6. For Delete at desired location");
            printf("\nPress 7. For Display the list");
            printf("\nEnter Choice= ");
            scanf("%d",&choice);
            printf("\n");
            switch(choice)
            {
                case 1: insertFront();
                        break;
                case 2: insertEnd();
                        break;
                case 3: insertPosition();
                        break;
                case 4: deleteFirst();
                        break;
                case 5: deleteEnd();
                        break;
                case 6: deletePosition();
                        break;
                case 7: display();
                        break;
                default: printf("Invalid Choice");
                        break;
            }
        }
        return 0;
}
```

Output:-



```
jaymin valaki: MAO75

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 1

Enter value= 1

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 1

Enter value= 2

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 3

Elements= 1 2

Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= 2


Press 1. For Insert data in Queue
Press 2. For Delete data From Queue
Press 3. For Display Elements Of Queue
Press 4. For Exit
Enter Your Choice= SKilled
```

4)Write program for all operations of circular doubly linked list.

- Inserting Node – as First Node, at desired location, as Last Node
- Deleting Node – at First, at Last, Specific Node
- Display List

.

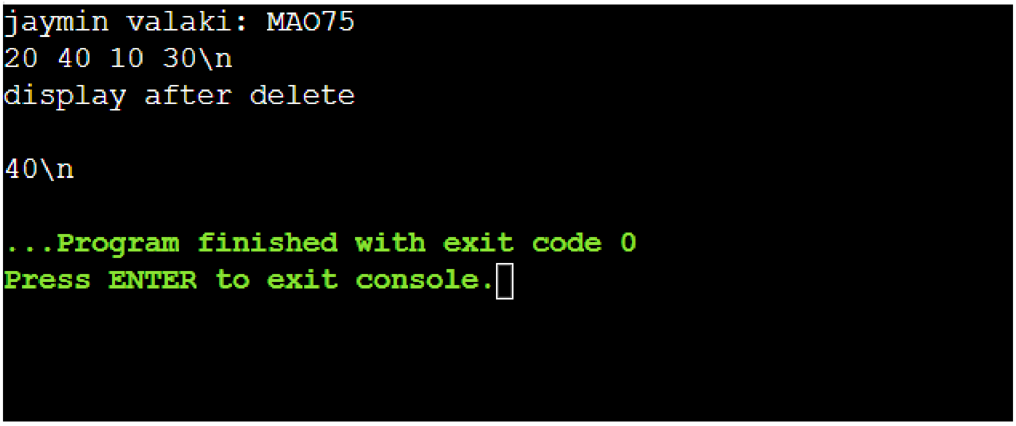
Source Code:-

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

```
typedef struct node {
    int data;
    struct node *next;
    struct node *prev;
} Node;

Node *head = NULL;

void insertFirst(int data) {
    Node *newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    if (head == NULL) {
        head = newNode;
        newNode->next = head;
        newNode->prev = head;
    } else {
        Node *last = head->prev;
        newNode->next = head;
        newNode->prev = last;
        last->next = newNode;
        head->prev = newNode;
        head = newNode;
    }
}

void insertLast(int data) {
    Node *newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    if (head == NULL) {
        head = newNode;
        newNode->next = head;
        newNode->prev = head;
    } else {
        Node *last = head->prev;
        newNode->next = head;
        newNode->prev = last;
        last->next = newNode;
        head->prev = newNode;
    }
```

```
}

void insertAt(int data, int position) {
    Node *newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;

    if (position == 1) {
        insertFirst(data);
    } else {
        Node *temp = head;
        for (int i=1; i<position-1; i++) {
            temp=temp->next;
            if (temp == head) {
                printf("Invalid position\\n");
                return;
            }
        }
        Node *nextNode=temp->next;

        temp->next=newNode;
        nextNode->prev=newNode;

        newNode->prev=temp;
        newNode->next=nextNode;

    }
}

void deleteFirst() {
    if (head == NULL) {
        printf("List is empty\\n");
    } else if (head->next == head) {
        free(head);
        head=NULL;
    } else {
        Node *last=head->prev;

        last->next=head->next;
        head->next->prev=last;
        free(head);
        head=last->next;

    }
}

void deleteLast() {
    if (head == NULL) {
        printf("List is empty\\n");
    } else if (head->next == head) {
        free(head);
        head=NULL;
    } else {
      Node *last=head->prev;

      last->prev->next=head;
      head->prev=last->prev;

      free(last);
  }
}

void deleteAt(int position) {

  if (position == 1) {
      deleteFirst();
  } else {

      Node *temp=head;

      for (int i=1; i<position; i++) {

          temp=temp->next;

          if (temp == head) {
              printf("Invalid position\\n");
              return;
          }
      }
      temp->prev->next=temp->next;
      temp->next->prev=temp->prev;

      free(temp);

  }
}

void displayList() {

  if (head == NULL) {
      printf("List is empty\\n");
  } else {

      Node *temp=head;

      while(temp->next != head) {

          printf("%d ", temp->data);
          temp=temp->next;

      }

      printf("%d\\n", temp->data);

  }

}

int main() {
 printf("jaymin valaki: MAO75\n");
  insertFirst(10);
  insertFirst(20);
  insertLast(30);
  insertAt(40,2);
  displayList();
  printf("\ndisplay after delete");

  deleteFirst();
  deleteLast();
  printf("\n");
  deleteAt(2);
  printf("\n");
  displayList();

  return 0;

}
```

OUTPUT:



```
jaymin valaki: MA075
20 40 10 30\n
display after delete

40\n

...Program finished with exit code 0
Press ENTER to exit console.
```