WAP to Implement Singly Linked List with following operations a) 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.

WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

```
#include<stdio.h>
#include<malloc.h>
struct node
{
        int info;
        struct node *next;
};

typedef struct node *NODE;

NODE getNode()
{
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        if(temp == NULL)
        {
                printf("Memory is full\n");
                return NULL;
        }
        return temp;
}

void freeNode(NODE temp)
{
        free(temp);
}

NODE insertFront(NODE first)
{
```

```
        int num;
        printf("Enter the number\n");
        scanf("%d",&num);
        NODE temp;
        temp = getNode();
        temp->info = num;
        temp->next = NULL;
        if(first==NULL)
        {
                return temp;
        }
        temp->next = first;
        first = temp;
        return first;
}

NODE insertRear(NODE first)
{
        int num;
        printf("Enter the number\n");
        scanf("%d",&num);
        NODE curr,temp;
        temp = getNode();
        temp->info = num;
        temp->next = NULL;
        if(first == NULL)
        {
                return temp;
        }
        curr = first;
        while(curr->next != NULL)
        {
                curr=curr->next;
        }
        curr->next = temp;
        return first;
}
```

```
NODE insertPos(NODE first,int pos)
{
	int num;
	printf("Enter the number\n");
	scanf("%d",&num);
	NODE temp,curr,prev;
	int count=1;
	temp = getNode();
	temp->info = num;
	temp->next = NULL;
	if(pos==count)
	{
		return temp;
	}
	curr = first;
	prev = NULL;
	while(curr!=NULL)
	{
		if(count==pos)
		{
			break;
		}
		prev=curr;
		curr=curr->next;
		count++;
	}
	if(curr==NULL)
	{
		printf("Enterred position is more than length\n");
		return first;
	}
	temp->next = prev->next;
	prev->next = temp;
	return first;
}
```

```
void display(NODE first)
{
	if(first == NULL)
	{
		printf("List is empty\n");
		return;
	}
	NODE curr = first;
	while(curr!=NULL
	){
		printf("%d ",curr->info);
		curr = curr->next;
	}
	printf("\n");
}

NODE deleteFront(NODE first)
{
	NODE curr;
	if(first == NULL)
	{
		printf("List is empty\n");
		return first;
	}
	if(first->next == NULL)
	{
		printf("Deleted element = %d\n",first->info);
		freeNode(first);
		return NULL;
	}
	curr=first;
	curr=curr->next;
	printf("Deleted element = %d\n",first->info);
	freeNode(first);
	return curr;
}
```

```
NODE deleteRear(NODE first)
{
	NODE curr,prev=NULL;
	if(first == NULL)
	{
		printf("List is empty\n");
		return first;
	}
	if(first->next == NULL)
	{
		printf("Deleted element = %d\n",first->info);
		freeNode(first);
		return NULL;
	}
	curr = first;
	while(curr->next!=NULL)
	{
		prev = curr;
		curr=curr->next;
	}
	prev->next = NULL;
	printf("Deleted element = %d\n",curr->info);
	freeNode(curr);
	return first;
}

NODE deletePos(NODE first,int pos)
{
	NODE curr,prev;
	if(first==NULL)
	{
		printf("List is empty\n");
		return first;
	}
	int count=1;
	if(pos==count)
	{
```

```
                printf("Deleted element = %d\n",first->info);
                curr = first;
                curr = curr->next;
                freeNode(first);
                return curr;
        }
        curr = first;
        prev = NULL;
        while(curr!=NULL)
        {
                if(count==pos)
                {
                        break;
                }
                prev = curr;
                curr = curr->next;
                count++;
        }
        if(curr==NULL)
        {
                printf("Enterred position is greater than length\n");
                return first;
        }
        prev->next = curr->next;
        printf("Deleted element = %d\n",curr->info);
        freeNode(curr);
        return first;
}

int main()
{
        int ch;
        int pos;
        NODE first = NULL;
        while(1)
        {
```

```
		printf("Enter the option\n1-insertFront\n2-insertRear\n3-
insertPosition\n4-deletedFront\n5-deletedRear\n6-deletePosition\n7-
display\n8-exit\n");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
				first = insertFront(first);
				break;
			case 2:
				first = insertRear(first);
				break;
			case 3:
				printf("Enter the position for insertion\n");
				scanf("%d",&pos);
				first = insertPos(first,pos);
				break;
			case 4:
				first = deleteFront(first);
				break;
			case 5:
				first = deleteRear(first);
				break;
			case 6:
				printf("Enter the position for deletion\n");
				scanf("%d",&pos);
				first = deletePos(first,pos);
				break;
			case 7:
				display(first);
				break;
			case 8:
			return 0;
			break;
		}
	}
}
```

```
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
1
Enter the number
20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
1
Enter the number
30
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit

7
30 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
2
Enter the number
10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
30 20 10
Enter the option
1-insertFront
2-insertRear
```

```
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
1
Enter the number
50
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
50 30 20 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
3
Enter the position for insertion
2
Enter the number
40
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
50 40 30 20 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
```

```
50 40 30 20 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
4
Deleted element = 50
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
5
Deleted element = 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
40 30 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
6
Enter the position for deletion
2
Deleted element = 30
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
40 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
4
Deleted element = 40
Enter the option
1-insertFront
2-insertRear
3-insertPosition
```

```
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
5
Deleted element = 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
4
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
5
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
8

--------------------------------
Process exited after 181.2 seconds with return value 0
Press any key to continue . . .
```