NAME - AKRITI CHOUDHARY
ROLL NO. - 2005776
CLASS - B14
DSA LAB ENDSEM

Question 1)WAP to reverse the first m elements of a linked list of n nodes.

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

struct node
{
   int num;
   struct node *next;
};

void createList(struct node **);
void reverseList(struct node **, int);
void releaseNode(struct node **);
void displayList(struct node *);

int main()
{
   struct node *p = NULL;
   int n;

   puts("Enter the data : ");
   createList(&p);
   puts("Displaying the nodes : ");
   displayList(p);
   puts("Enter the number of nodes to be reversed: ");
   scanf("%d", &n);

   if (n > 1)
   {
      reverseList(&p, n - 2);
   }
   puts("The reversed list(first m elements of n nodes): ");
   displayList(p);

   releaseNode(&p);

   return 0;
}

void reverseList(struct node **head, int n)
{
   struct node *p, *q, *r, *rear;
```

```
    p = q = r = *head;
    if (n == 0)
    {
        q = q->next;
        p->next = q->next;
        q->next = p;
        *head = q;
    }
    else
    {
        p = p->next->next;
        q = q->next;
        r->next = NULL;
        rear = r;
        q->next = r;

        while (n > 0 && p != NULL)
        {
            r = q;
            q = p;
            p = p->next;
            q->next = r;
            n--;
        }
        *head = q;
        rear->next = p;
    }
}

void createList(struct node **head)
{
    int c, ch;
    struct node *temp, *rear;

    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        puts("Enter 1 - to continue and 0 - to stop creation of the list");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
}

void displayList(struct node *p)
```

```
{
   while (p != NULL)
   {
      printf("%d\t", p->num);
      p = p->next;
   }
   printf("\n");
}

void releaseNode(struct node **head)
{
   struct node *temp = *head;
   *head = (*head)->next;
   while ((*head) != NULL)
   {
      free(temp);
      temp = *head;
      (*head) = (*head)->next;
   }
}
```

```
PS C:\Users\KIIT\OneDrive\Desktop\test> g++ reverse.c -oreverse
PS C:\Users\KIIT\OneDrive\Desktop\test> ./reverse
Enter the data :
Enter number: 45
Enter 1 - to continue and 0 - to stop creation of the list
1
Enter number: 67
Enter 1 - to continue and 0 - to stop creation of the list
1
Enter number: 34
Enter 1 - to continue and 0 - to stop creation of the list
1
Enter number: 2
Enter 1 - to continue and 0 - to stop creation of the list
1
Enter number: 9
Enter 1 - to continue and 0 - to stop creation of the list
1
Enter number: 56
Enter 1 - to continue and 0 - to stop creation of the list
0

Displaying the nodes :
45      67      34      2       9       56
Enter the number of nodes to be reversed:
3
The reversed list(first m elements of n nodes):
```

```
Displaying the nodes :
45      67      34      2       9       56
Enter the number of nodes to be reversed:
3
The reversed list(first m elements of n nodes):
34      67      45      2       9       56
PS C:\Users\KIIT\OneDrive\Desktop\test>
```

Question 2)WAP to sort an array of n integers in an ascending order by using quicksort.

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

int *arr;
int n;

void display()
{
   printf("Elements of the array : \n");

   for (int i = 0; i < n; i++)
   {
      printf("%d ", arr[i]);
   }
   printf("\n");
}

int partition(int *A, int low, int high)
{
   int pivot = A[low];
   int i = low + 1;
   int j = high;
   int temp;

   do
   {
      while (A[i] <= pivot)
      {
         i++;
      }

      while (A[j] > pivot)
      {
         j--;
      }

      if (i < j)
      {
         temp = A[i];
         A[i] = A[j];
         A[j] = temp;
```

```
        }
    } while (i < j);

    temp = A[low];
    A[low] = A[j];
    A[j] = temp;
    return j;
}

void quickSort(int *A, int low, int high)
{
    int partitionIndex;

    if (low < high)
    {
        partitionIndex = partition(A, low, high);
        quickSort(A, low, partitionIndex - 1);
        quickSort(A, partitionIndex + 1, high);
    }
}

int main()
{

    printf("Enter the size of the array : \n");
    scanf("%d", &n);

    arr = (int *)malloc(n * sizeof(int));

    for (int i = 0; i < n; i++)
    {
        printf("Enter %d element : \n", i + 1);
        scanf("%d", &arr[i]);
    }

    quickSort(arr, 0, n - 1);
    display();

    return 0;
}
```

```
PS C:\Users\KIIT\OneDrive\Desktop\test> ./quicksort
Enter the size of the array :
7
Enter 1 element :
12
Enter 2 element :
90
Enter 3 element :
1
Enter 4 element :
45
Enter 5 element :
-2
Enter 6 element :
65
Enter 7 element :
5
Elements of the array :
-2 1 5 12 45 65 90
PS C:\Users\KIIT\OneDrive\Desktop\test>
```