NAME: SUSMITA PRATIHAR
ENROLL : 12019002001087 SEC: B CSE

1. Write a C program to implement quick sort using divide and conquer method. (for array and linked list both)

ANS:- ALGORITHM:

a) Choosing a pivot
-To partition the list we first choose a pivot element

b) Partitioning

-Then we partition the elements so that all those with values less than pivot are placed on the left side and the higher vale on the right.
-Check if the current element is less than the pivot.
- If lesser replace it with the current element and move the wall up one position
- else move the pivot element to current element and vice versa

c) Recur
- Repeat the same partitioning step unless all elements are sorted

main.c

```c
#include<stdio.h>
void quicksort(int number[25],int first,int last){
   int i, j, pivot, temp;

   if(first<last){
      pivot=first;
      i=first;
      j=last;

      while(i<j){
         while(number[i]<=number[pivot]&&i<last)
            i++;
         while(number[j]>number[pivot])
            j--;
         if(i<j){
            temp=number[i];
            number[i]=number[j];
            number[j]=temp;
         }
      }

      temp=number[pivot];
      number[pivot]=number[j];
      number[j]=temp;
      quicksort(number,first,j-1);
      quicksort(number,j+1,last);

   }
}
int main(){
   int i, count, number[25];

   printf("How many elements are u going to enter?: ");
   scanf("%d",&count);

   printf("Enter %d elements: ", count);
   for(i=0;i<count;i++)
      scanf("%d",&number[i]);

   quicksort(number,0,count-1);

   printf("Order of Sorted elements: ");
   for(i=0;i<count;i++)
      printf(" %d",number[i]);

   return 0;
}
```

```
How many elements are u going to enter?: 4
Enter 4 elements: 11
2
1
21
Order of Sorted elements:  1 2 11 21

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

main.c

```c
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
    struct Node *prev;
};

void swap ( int* a, int* b )
{ int t = *a; *a = *b; *b = t; }

struct Node *lastNode(struct Node *root)
{
    while (root && root->next)
        root = root->next;
    return root;
}

struct Node* partition(struct Node *l, struct Node *h)
{
    int x = h->data;
    struct Node *i = l->prev;
    for (struct Node *j = l; j != h; j = j->next)
    {
        if (j->data <= x)
        {
            i = (i == NULL) ? l : i->next;

            swap(&(i->data), &(j->data));
        }
    }
    i = (i == NULL) ? l : i->next;
    swap(&(i->data), &(h->data));
    return i;
}

void _quickSort(struct Node* l, struct Node *h)
{
    if (h != NULL && l != h && l != h->next)
    {
        struct Node *p = partition(l, h);
        _quickSort(l, p->prev);
        _quickSort(p->next, h);
    }
}

void quickSort(struct Node *head)
{
    struct Node *h = lastNode(head);
    _quickSort(head, h);
}

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

void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*)
                malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->prev = NULL;
    new_node->next = (*head_ref);
    if ((*head_ref) != NULL) (*head_ref)->prev = new_node ;
    (*head_ref) = new_node;
}
int main(int argc, char **argv)
{
    struct Node *a = NULL;
    push(&a, 5);
    push(&a, 21);
    push(&a, 14);
    push(&a, 80);
    push(&a, 22);

    printf("Linked List before sorting \n");
    printList(a);

    quickSort(a);

    printf("Linked List after sorting \n");
    printList(a);

    return 0;
}
```

```
Linked List before sorting
22 80 14 21 5
Linked List after sorting
5 14 21 22 80


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