```
/*Assignment 7:-
Write a program in c to perform ARKPIan Graph Sort on a given set of data.*/

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

/*node structure*/
struct node
{
    int data,pos,src,ntrmdt,sink;
    struct node *next,*spar;
};

/*creating a node*/
struct node *getnode(int x,int i)
{
    struct node *new1;
    new1=(struct node *)malloc(sizeof(struct node));
    new1->data=x;
    new1->pos=i;
    new1->src=0;
    new1->ntrmdt=0;
    new1->sink=0;
    new1->next=NULL;
    new1->spar=NULL;
    return(new1);
}

/*declaring global variables*/
struct node *ihead=NULL,*fhead=NULL;
int n=0;

void main()
{
    void insert();
    void partition();
    void adjust();
    void arkpian_sort();
    void display();
    insert();
    printf("\nThe list before sorting:-\n");
    display();
    arkpian_sort();
    printf("\nThe list after sorting:-\n");
    display();
}

/*function to insert the elements of the list on which sorting is to be performed*/
void insert()
{
    int x;
    struct node *new1,*ptr;
    do
    {
        printf("Enter element :- ");
        scanf("%d",&x);
        new1=getnode(x,++n);
        if(ihead == NULL)
            ihead=new1;
        else
        {
            ptr=ihead;
            while(ptr->next != NULL)
                ptr=ptr->next;
            ptr->next=new1;
        }
        printf("Do you want to enter any more element?(0=YES,1=NO) :- ");
        scanf("%d",&x);
    }while(x==1);
}

/*function to perform the initial partitioning of the given list*/
void partition()
{
    struct node *ptr1,*ptr2,*ptr3,*new1;
    ptr1=ihead;
    while(ptr1 != NULL)
    {
        ptr2=ptr1->next;
        while(ptr2 != NULL)
        {
```

```
            if(ptr1->data <= ptr2->data)
            {
                new1=getnode(ptr2->data,ptr2->pos);
                ptr3=ptr1;
                while(ptr3->spar != NULL)
                    ptr3=ptr3->spar;
                ptr3->spar=new1;
            }
            ptr2=ptr2->next;
        }
        ptr1=ptr1->next;
    }
}

/*adjusting the existing list after the minimum element key is deleted*/
void adjust()
{
    struct node *ptr1,*ptr2,*ptr3;
    int has;
    /*loop for determining if the nodes are source,sink or intermidiate*/
    ptr1=ihead;
    while(ptr1 != NULL)
    {
        has=0;
        ptr2=ihead;
        while(ptr2 != NULL)
        {
            ptr3=ptr2->spar;
            while(ptr3 != NULL)
            {
                if(ptr3->data == ptr1->data && ptr3->pos == ptr1->pos)
                    has++;
                ptr3=ptr3->spar;
            }
            ptr2=ptr2->next;
        }
        if((ptr1->spar == NULL && has == 0) || has == 0)
        {
            ptr1->sink=0;
            ptr1->ntrmdt=0;
            ptr1->src=1;
        }
        else if(ptr1->spar == NULL)
        {
            ptr1->src=0;
            ptr1->sink=1;
            ptr1->ntrmdt=0;
        }
        else
        {
            ptr1->ntrmdt=1;
            ptr1->src=0;
            ptr1->sink=0;
        }
        ptr1=ptr1->next;
    }
}

/*function to perform the sorting*/
void arkpian_sort()
{
    struct node *ptr1,*ptr2,*min;
    /*calling the partition function to partition the given list initially*/
    partition();
    /*loop for performing the sorting algorithm*/
    while(ihead != NULL)
    {
        /*adjusting the given list after the minimum key is deleted*/
        adjust();
        ptr1=ihead;
        while(ptr1 != NULL)
        {
            if(ptr1->src == 1)
            {
                min=ptr1;
                break;
            }
            ptr1=ptr1->next;
        }
        ptr1=ptr1->next;
        while(ptr1 != NULL)
        {
```

```
            if(min->data >= ptr1->data && ptr1->src == 1)
                min=ptr1;
            ptr1=ptr1->next;
        }
        if(ihead == min)
            ihead=ihead->next;
        else
        {
            ptr1=ihead;
            while(ptr1->next != min)
                ptr1=ptr1->next;
            ptr1->next=min->next;
        }
        ptr1=min->spar;
        min->src=0;
        min->sink=0;
        min->ntrmdt=0;
        min->spar=NULL;
        min->next=NULL;
        while(ptr1 != NULL)
        {
            ptr2=ptr1;
            ptr1=ptr1->spar;
            free(ptr2);
        }
        if(fhead == NULL)
            fhead=min;
        else
        {
            ptr1=fhead;
            while(ptr1->next != NULL)
                ptr1=ptr1->next;
            ptr1->next=min;
            if(ihead->next == NULL)
            {
                min->next=ihead;
                ihead=ihead->next;
            }
        }
    }
}

/*displaying the initial list and the sorted list*/
void display()
{
    struct node *ptr;
    if(ihead != NULL)
    {
        ptr=ihead;
        printf("\n");
        while(ptr != NULL)
        {
            printf("%d(%d)\t",ptr->data,ptr->pos);
            ptr=ptr->next;
        }
    }
    if(fhead != NULL)
    {
        ptr=fhead;
        printf("\n");
        while(ptr != NULL)
        {
            printf("%d(%d)\t",ptr->data,ptr->pos);
            ptr=ptr->next;
        }
    }
    printf("\n");
}
```