/*
Q3. Write a C program to create a linked list P, then write a 'C' function named split to create two linked lists Q & R from P So that Q contains all elements in odd positions of P and R contains the remaining elements. Finally print both linked lists i.e. Q and R.
Name-Deepanshu Negi
Section-C2
Roll no-24
Course-Btech(CSE)
*/

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node*next;
};typedef struct node node;

void insert(node**P , node**tail , int val){
   node*temp1=(node*)malloc(sizeof(node));
   temp1->data=val;
   temp1->next=NULL;
   if(*P==NULL){
      *P=temp1;
      *tail=temp1;
   }else{
      (*tail)->next=temp1;
      *tail=temp1;
   }
}
void split_list(node*P,node**Q,node**R){
   *Q=NULL,*R=NULL;
   node*tail1=NULL,*tail2=NULL;
   int i=1;
   while(P!=NULL){
      if(i%2==0){
         insert(R,&tail2,P->data);
      }else{
         insert(Q,&tail1,P->data);
      }
      i++;
      P=P->next;
   }
}
void print(node*P){
   if(P==NULL){
      printf("List Empty\n");
      return;
   }
```

```
   while(P!=NULL){
      printf("%d ",P->data);
      P=P->next;
   }
}
int main(){
   node*P=NULL;
   node*Q=NULL;
   node*R=NULL;
   node*tail=NULL;
   int ch,val;
   printf("OPERATIONS\nPress 1 To Insert\nPress 2 To Split\nPress 3 To Print Q\nPress 4 To
Print R\nPress 5 To Exit\n");
   while(1){
      scanf("%d",&ch);
      switch(ch){
         case 1:
         printf("Enter the value to insert:");
         scanf("%d",&val);
         insert(&P,&tail,val);
         print(P);
         printf("\n");
         break;

         case 2:
         split_list(P,&Q,&R);
         printf("List Splitted\n");
         break;

         case 3:
         printf("Odd Index List Q:");
         print(Q);
         printf("\n");
         break;

         case 4:
         printf("Even Index List R:");
         print(R);
         printf("\n");
         break;

         case 5:
         printf("EXIT");
         exit(0);

         default:
         printf("Invalid choice\n");
      } }
return 0;
}
```

OUTPUT:

```
PS C:\Deepanshu C> cd "c:\Deepanshu C\coding\TERM WORK\" ; if ($?) { gcc Q3.c -o Q3
} ; if ($?) { .\Q3 }
OPERATIONS
Press 1 To Insert
Press 2 To Split
Press 3 To Print Q
Press 4 To Print R
Press 5 To Exit
1
Enter the value to insert:28
28
1
Enter the value to insert:62
28 62
1
Enter the value to insert:44
28 62 44
1
Enter the value to insert:20
28 62 44 20
1
Enter the value to insert:32
28 62 44 20 32
1
Enter the value to insert:8
28 62 44 20 32 8
2
List Splitted
3
Odd Index List Q:28 44 32
4
Even Index List R:62 20 8
5
EXIT
```

/*
Q2. Write a C program to find union (of two linked lists) based on their information field that implements singly linked list (with information field Emp_Id and Name of employee for each node).
Name-Deepanshu Negi
Section-C2
Roll no-24
Course-Btech(CSE)
*/

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct node{
   struct node*next;
   int EmpID;
   char arr[100];
}node;
//create a newnode
node*createNode(int ID,char name[]){
   node*newnode=(node*)malloc(sizeof(node));
   strcpy(newnode->arr,name);
   newnode->EmpID=ID;
   newnode->next=NULL;
   return newnode;
}
int Check_Unique(node*head,node*head2){
   while(head!=NULL){
      if(head2->EmpID==head->EmpID){
         return 0;
      }
      head=head->next;
   }
   return 1;
}
//function to insert at beginning of list
node*Insert(node*head,int ID,char name[]){
   if(head==NULL){
      head=createNode(ID,name);
      return head;
   }
   node*temp=createNode(ID,name);
   temp->next=head;
   head=temp;
   return head;
}
//find union based on EmpID
node*Union(node*head1,node*head2){
```

```
    node*head=NULL;
    //copies the first list to resultant list
    while(head1!=NULL){
        head=Insert(head,head1->EmpID,head1->arr);
        head1=head1->next;
    }
    while(head2!=NULL){
        //checks if EmpID is already present in resultant list
        if(Check_Unique(head,head2)){
            head=Insert(head,head2->EmpID,head2->arr);
        }
        head2=head2->next;
    }
    return head;
}
void freeList(node*head){
    while(head!=NULL){
        node*temp=head;
        head=head->next;
        free(temp->arr);
        free(temp);
    }
}
void display(node*head){
    while(head!=NULL){
        printf("%d ",head->EmpID);
        printf("%s\n",head->arr);
        head=head->next;
    }
}
int main(){
    node*head1=NULL;
    node*head2=NULL;
    node*head3=NULL;
    int ch=0;
    printf("Press -1 to exit\n");
    printf("Enter first list\n");
    while(ch!=-1){
        char name[100];
        int ID;
        printf("Enter the EmpID:");
        scanf("%d",&ID);
        printf("Enter the name:");
        scanf("%s",name);
        head1=Insert(head1,ID,name);
        scanf("%d",&ch);
    }
    printf("Enter second linked list:\n");
    ch=0;
    while(ch!=-1){
```

```
        char name[100];
        int ID;
        printf("Enter the emp-ID:");
        scanf("%d",&ID);
        printf("Enter the name:");
        scanf("%s",name);
        head2=Insert(head2,ID,name);
        scanf("%d",&ch);
    }
    printf("First list:\n");
    display(head1);
    printf("Second list:\n");
    display(head2);
    head3=Union(head1,head2);
    printf("Union of list 1 and list 2:\n");
    display(head3);
    // Free the memory allocated for the linked lists
    freeList(head1);
    freeList(head2);
    freeList(head3);
    return 0;
}
```

OUTPUT:

```
PS C:\Deepanshu C> cd "c:\Deepanshu C\coding\TERM WORK\" ; if ($?) { gcc Q2.c -o Q2
} ; if ($?) { .\Q2 }
Press -1 to exit
Enter first list
Enter the EmpID:101
Enter the name:Deepanshu
1
Enter the EmpID:102
Enter the name:Ajay
-1
Enter second linked list:
Enter the emp-ID:103
Enter the name:Arpit
1
Enter the emp-ID:102
Enter the name:Rahul
1
Enter the emp-ID:104
Enter the name:Kashish
-1
First list:
102 Ajay
101 Deepanshu
Second list:
104 Kashish
102 Rahul
103 Arpit
Union of list 1 and list 2:
103 Arpit
104 Kashish
101 Deepanshu
102 Ajay
```

Q1. Write a C program to implement priority queue using doubly linked list (Priority depends on identity number. Small identity number has greater priority. If identity numbers are equal. Then FIFO rules are used with following functions,
1) insert 2) delete 3) display
Name-Deepanshu Negi
Section-C2
Roll no-24
Course-Btech(CSE)
*/

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>
//defining the structure of the priority queue
typedef struct node{
   int Id_no;
   int data;
   struct node*next;
   struct node*prev;
}node;
//create a new node
node*createnode(int val,int pri){
   node*temp=(node*)malloc(sizeof(node));
   temp->data=val;
   temp->Id_no=pri;
   temp->next=temp->prev=NULL;
   return temp;
}
// to enqueue based on Id_no
void enqueue(node**head,int val,int pri){
   if(*head==NULL){    //empty list
      *head=createnode(val,pri);
      return;
   }
   node*temp=*head;
   node*temp2=temp;
   //searching in a list at which place to insert newNode(based on Id_no)
   while(temp!=NULL && (temp)->Id_no <= pri){
      temp2=temp;
      temp=temp->next;
   }
   node*newQueue=createnode(val,pri);
   if(temp==*head){    //if inserting at head
      newQueue->next=*head;
      (*head)->prev=newQueue;
      *head=newQueue;
      return;
```

```
    }
    if(temp2->next==NULL){ //if inserting at tail
        temp2->next=newQueue;
        return;
    }
    //inserting at between
    newQueue->next=temp2->next;
    temp2->next=newQueue;
    newQueue->prev=temp2;
}
int dequeue(node**head){
    if(*head==NULL){
        printf("Queue empty\n");
        return INT_MIN;
    }
    node*temp=*head;     //dequeue the front element in list
    int val=temp->data;
    (*head)=(*head)->next;
    free(temp);
    return val;
}
void display(node*head){
    printf("Priority queue:\n");
    while(head!=NULL){
        printf("%d  %d\n",head->Id_no,head->data);
        head=head->next;
    }
}
int main(){
    node*head=NULL;
    int ch=1;
    printf("Press\n1:Enqueue\n2:Dequeue\n3:Exit\n");
    while(ch!=3){
        scanf("%d",&ch);
        switch(ch){
            case 1:{
            int Id_no,pri;
            printf("Enter the number:");
            scanf("%d",&Id_no);
            printf("Enter its priority:");
            scanf("%d",&pri);
            enqueue(&head,Id_no,pri);
            display(head);
            }
            break;

            case 2:{
            int el=dequeue(&head);
            if(el==INT_MIN){
                break;
```

```
            }
            printf("Element deleted:%d\n",el);
            display(head);
            }
            break;

            case 3:
            printf("Priority queue implemented\n");
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Deepanshu C> cd "c:\Deepanshu C\coding\TERM WORK\" ; if ($?) { gcc Q1.c -o Q1
} ; if ($?) { .\Q1 }
Press
1:Enqueue
2:Dequeue
3:Exit
1
Enter the number:20
Enter its priority:4
Priority queue:
4  20
1
Enter the number:66
Enter its priority:1
Priority queue:
1  66
4  20
1
Enter the number:35
Enter its priority:2
Priority queue:
1  66
2  35
4  20
1
Enter the number:100
Enter its priority:5
Priority queue:
1  66
2  35
4  20
5  100
2
Element deleted:66
Priority queue:
2  35
4  20
5  100
3
Priority queue implemented
```

/*
Q4. Write a C program to create a binary search tree and perform following operations:
1) Find node having smallest data in the BST.
2) Delete a node from the tree.
3) Find total number nodes having common parent.
4) Find height of a binary search tree
5) Count total numbers of nodes from right hand side of root node
Name-Deepanshu Negi
Section-C2
Roll no-24
Course-Btech(CSE)
*/

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
struct node{
   int key;
   struct node *left, *right;
};typedef struct node node;

// Function to create a new node
node *create(int item){
   node*temp=(node*)malloc(sizeof(node));
   temp->key=item;
   temp->left=NULL;
   temp->right=NULL;
   return temp;
}
// Function to insert node in BST
node *insert(node *root,int key){
   if(root==NULL)
      return create(key);
   if(root->key==key)
      return root;
   if(root->key > key)
      root->left=insert(root->left,key);
   if(root->key < key)
      root->right=insert(root->right,key);
   return root;
}
// Function to find successor
node *successor(node*root){
   node*curr=root;
   while(curr!=NULL && curr->left!=NULL){
      curr=curr->left;
   }
   return curr;
```

```
}
// Function to delete a node
node *delete(node *root,int key){
   if(root==NULL)
      return root;
   if(root->key > key)
      root->left=delete(root->left,key);
   else if(root->key < key)
      root->right=delete(root->right,key);
   else{
      if(root->left==NULL && root->right==NULL){
         free(root);
         return NULL;
      }
      if(root->left==NULL){
         node *temp = root->right;
         free(root);
         return temp;
      }
      if(root->right==NULL){
         node *temp=root->left;
         free(root);
         return temp;
      }
      node *temp=successor(root->right);
      root->key=temp->key;
      root->right=delete(root->right,temp->key);
      return root;
   }
}
// Display the nodes
void inorder(node *root){
   if(root==NULL)
   return;
   inorder(root->left);
   printf("%d ", root->key);
   inorder(root->right);
}
// Find the minimum value
void minimum(node *root){
   node *curr=root;
   while(curr!=NULL && curr->left!=NULL){
      curr=curr->left;
   }
   printf("%d",curr->key);
}
// Counting Common Parents
int count_commonparent(node *root){
   if (root==NULL){
      return 0;
```

```
    }
    int c=0;
    if(root->left!=NULL && root->right!= NULL){
        c=2;
    }
    c=c + count_commonparent(root->left);
    c=c + count_commonparent(root->right);
    return c;
}
// Counting right sub-tree nodes
int count_rightnodes(node *root){
    if (root==NULL){
        return 0;
    }
    int count=0;
    if (root->right!=NULL){
        count=1;
    }
    count += count_rightnodes(root->left);
    count += count_rightnodes(root->right);
    return count;
}
// find maximum element
int max(int left,int right){
    if(left>right){
        return left;
    }
    return right;
}
// Height of BST
int height(node *root){
    if(root==NULL)
        return -1;
    int lh=height(root->left);
    int rh=height(root->right);
    return(lh > rh ? lh : rh)+1;
}
int main(){
    node*root=NULL;
    int ch,n,value;
    printf("1.Insert a node\n2.Delete a node\n3.Height of a tree\n4.Total number of nodes of
common parent\n5.Smallest node\n6.Total number of nodes from right hand side\n7.Display
tree\n8.Exit");
    do{
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch){
            case 1:
            printf("Enter no.of nodes:");
            scanf("%d",&n);
```

```
            printf("Enter nodes:");
            for(int i=0;i<n; i++){
                scanf("%d",&value);
                root=insert(root,value);
            }
            break;

            case 2:
            printf("Enter the node to delete: ");
            scanf("%d",&value);
            root=delete (root,value);
            break;

            case 3:
            printf("Height of the tree is: %d",height(root));
            break;

            case 4:
             printf("Total number of nodes having common parent are:
            %d",count_commonparent(root));
            break;

            case 5:
            printf("Smallest node is: ");
            minimum(root);
            break;

            case 6:
            printf("Total number of nodes from right side are: %d",count_rightnodes(root));
            break;

            case 7:
            printf("BST Tree is:");
            inorder(root);
            break;

            case 8:
            printf("Exit");
            break;
        }
    } while(ch!=8);
    return 0;
}
```

OUTPUT:

```
PS C:\Deepanshu C> cd "c:\Deepanshu C\coding\TERM WORK\" ; if ($?) { gcc Q4.c -o Q4
} ; if ($?) { .\Q4 }
1.Insert a node
2.Delete a node
3.Height of a tree
4.Total number of nodes of common parent
5.Smallest node
6.Total number of nodes from right hand side
7.Display tree
8.Exit
Enter your choice:1
Enter no.of nodes:7
Enter nodes:20 55 39 4 19 43 60

Enter your choice:7
BST Tree is:4 19 20 39 43 55 60
Enter your choice:3
Height of the tree is: 3
Enter your choice:2
Enter the node to delete: 20

Enter your choice:7
BST Tree is:4 19 39 43 55 60
Enter your choice:6
Total number of nodes from right side are: 3
Enter your choice:8
Exit
```

/*
Q6. Write a C program to sort N names (as a string) given by user in an array, using Quick sort technique.
Name-Deepanshu Negi
Section-C2
Roll no-24
Course-Btech(CSE)
*/
SOURCE CODE:

```
#include <stdio.h>
#include <string.h>
#define MAX 100 // Maximum number of names
#define len 100 // Maximum length of each name

// Function to swap
void swap(char arr[][len], int i, int j){
    char temp[len];
    strcpy(temp, arr[i]);
    strcpy(arr[i], arr[j]);
    strcpy(arr[j], temp);
}

// Partition function for Quick Sort
int partition(char arr[][len], int low, int high){
    char pivot[len];
    strcpy(pivot,arr[high]); // Pivot is the last element
    int i=low - 1;
    int j=low;

    while(j < high){
        if(strcmp(arr[j],pivot) < 0){
            i++;
            swap(arr,i,j);
        }
        j++;
    }
    swap(arr,i + 1,high);
    return i + 1; // Pivot element index
}

// Quick Sort function
void quickSort(char arr[][len], int low, int high){
    if(low < high){
        int pi=partition(arr,low,high);

        quickSort(arr,low,pi - 1);  // left side
        quickSort(arr,pi + 1,high); // right side
    }
}
```

```
int main(){
    char names[MAX][len];
    int n;
    printf("Enter the number of names: ");
    scanf("%d", &n);

    //Input names
    printf("Enter %d names:\n", n);
    for(int i = 0; i < n; i++){
        scanf("%s", names[i]);
    }

    //Sort names using Quick Sort
    quickSort(names, 0, n - 1);

    //Display sorted names
    printf("\nSorted names:\n");
    for(int i = 0; i < n; i++){
        printf("%s\n", names[i]);
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Deepanshu C> cd "c:\Deepanshu C\coding\TERM WORK\" ; if ($?) { gcc Q6.c -o Q6 } ; if ($?) { .\Q6 }
Enter the number of names: 6
Enter 6 names:
Deepanshu
Tarun
Ajay
Kajal
Abhishek
Sam

Sorted names:
Abhishek
Ajay
Deepanshu
Kajal
Sam
Tarun
```