NR CLASSES LLP
THE TEACHING EXCELLENCE

SY BCS Data Structure - I

Solved Practical Slips

2022 -23

Solution Credit goes to

Anurag,Harshad,Shravani,Mahendra,Hritik,Samrudhhi,Shubham,Anish

www.nrclassespune.com | www.bcsbca.com

Slip 1_1: .Implement a list library (doublylist.h) for a doubly linked list of integers with the create,

display operations. Write a menu driven program to call these operations.

Solution:

Header File : doublylist.h

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

struct node {    struct node *prev;    int data;
struct node *next;
};

struct node *f;

void create()
{
   int n,i;
   struct node *s;
   printf("enter number of nodes needed : ");
   scanf("%d",&n);
  f  =(struct node *)malloc(sizeof(struct node));
   printf("enter data : ");
   scanf("%d",&f->data);
   f->prev=NULL;
   s=f;
   for(i=1;i<n;i++)
   {
      s->next=(struct node *)malloc(sizeof(struct node));      s=s->next;      printf("enter data :");
scanf("%d",&s->data);
   }
   s->next= NULL;
}
void display()
{
   struct node *s;    for(s=f;s!=NULL;s=s->next)
   {
      printf(" %d ->  ",s->data);
   }
}
```

Program File

```
#include <stdio.h>
#include "doublylist.h"
 int main() {  int ch;    do
         {
   printf("\n1.create\n2.display\n0.exit");    printf("enter choice :");
scanf("%d",&ch);    switch (ch)    {     case 1: create();
                     break;
    case 2: display();
                     break;
    case 0:
break;
```

```
         default:
      default:printf("invalid choice ");
                     break;
   }
   }while(ch!=0);
}
```

Slip 1_2, Slip 13_2 : Write a program that sorts the elements of linked list using any of sorting technique / Sort linked list using bubble sort

Solution:

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

struct node
{        int data;           struct node *next;
};
struct node *f;  void  create()  {          int i,n;       struct node *s;
printf("\nEnter no of nodes ");    scanf("%d",&n);
   f=(struct node *)malloc(sizeof(struct node));    printf("\n Enter node  ");
scanf("%d",&f->data);    s=f;



         for(i=1;i<n;i++)              {
                     s->next=(struct node *)malloc(sizeof(struct node));
                     s=s->next;
                     printf("\n Enter node  ");
                     scanf("%d",&s->data);
         }
         s->next=NULL;

}
void display()
{   struct node *s;
   for(s=f;s!=NULL;s=s->next)
         {           printf("\t %d ->",s->data);
         }
} void sort()
{        struct node *p,*q;          int temp;
         for(p=f;p!=NULL;p=p->next)
         {
                     for(q=p->next;q!=NULL;q=q->next)
```

```
                     {
                                                 if(p->data > q->data)
                                                 {           temp = p->data;
                                                             p->data = q->data;
                                                             q->data = temp;

                                                 }

                     }
         }
} main()
{
         create();
         printf("\n Link list is : ");
         display();
         printf("\n After sorting Link list is = ");
         sort();
         display();
}
```

Slip 2_1: Implement a list library (singlylist.h) for a singly linked list of integer with the operations create, display. Write a menu driven program to call these operations

Solution:

Header File : singlylist.h

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


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

struct node *f;
 void create() {    int n,i;    struct node *s;
   printf("enter number of nodes needed : ");    scanf("%d",&n);
   f=(struct node *)malloc(sizeof(struct node));    printf("enter data : ");
scanf("%d",&f->data);    s=f;    for(i=1;i<n;i++)
   {
      s->next=(struct node *)malloc(sizeof(struct node));      s=s->next;      printf("enter data :");
scanf("%d",&s->data);
   }    s->next= NULL;
} void display() {    struct
node *s;
for(s=f;s!=NULL;s=s-
>next)
   {
      printf("%d ->",s-                                                  >data);
   }
}
```

Program File :

```
#include <stdio.h>
#include "singlylist.h"
 main() {  int ch;    do{
   printf("\n1.create\n2.display\n3.exit");    printf("\nenter choice :");
scanf("%d",&ch);    switch (ch)    {    case 1: create();          break;    case 2:
display();          break;    case 3: break;
   default: printf("invalid input");
   }
   }while(ch!=3);
}
```

Slip 2_2 ,Slip 8_2, Slip 17_2: Write a program that copies the contents of one stack into another. Use stack library to perform basic stack operations. The order of two stacks must be identical.(Hint: Use a temporary stack to preserve the order).

```
#include <stdio.h>
 char s[20]; int top;  void
init() {    top=-1;
}  int isempty(){
if(top==-1)     return 1;
else     return 0;
}
int isfull() {    if(top==19)                                                   return 1;    else
return 0;
}
void push(char ch)
{
   if(isfull()==1)    printf("stack is full");    else    {
top++;       s[top]=ch;
   }
}  char pop() {    char ch;    if(isempty()==1)    printf("stack is
empty");    else    {       ch=s[top];       top--;    }   return ch;
}



 main()  {    int i,k=0;    char temp[20];    init();    char str[20];
printf("enter    string       ");                  scanf("%s",str);
for(i=0;str[i]!='\0';i++)
   {
      push(str[i]);
   }
   while(!isempty())
   {      temp[k]=pop();       k++;
   }
   temp[k]='\0';
```

```
  //Again String push into stack
   printf("second string :");
   for(i=0;temp[i]!='\0';i++)
   {
      push(temp[i]);
   }

   while(!isempty())
   {
      printf("%c",pop());
   }
 }
```

Slip 3_1 : Sort a random array of n integers (accept the value of n from user) in ascending order by using insertion sort algorithm.

Solution :

```
/* Insertion sort on random nos */
#include<stdio.h> int main() {
   int a[10],i,j,n,key;    printf("Enter how many numbers: ");
scanf("%d",&n);
      for(i=0; i<n; i++)    {       a[i]=rand()%100;
   }
   printf("\n Before sorting array is ");
   for(i=0;i<n;i++)    {  printf("%d ",a[i]);
   }    for(i=1; i<n; i++)    {       key = a[i];       for(j=i-1;
j>=0; j--)
      {          if(a[j] > key)          {
            a[j+1]=a[j];
         }          else             break;       }
a[j+1]=key;
   }       printf("\nAfter sort array is: ");    for(i=0; i<n; i++)
   {
      printf("%d ",a[i]);
   }
}
```

Slip 3_2 : Write a C program to evaluate postfix expression.

Slip 16_2 : A postfix expression of the form ab+cd-*ab/ is to be evaluated after accepting the values of a, b, c and d. Formulate the problem and write a C program to solve the problem by using stack.

Solution :

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

```
#include<string.h>  char s[20]; int top;  void
init()
{
         top=-1;
}
int isempty()
{        if(top==-1)
                     return 1;
         else return 0;
}

int isfull()
{        if(top==19)
                     return 1;           else
                     return 0;
}
void push(char data)
{
         if(isfull()==1)                 printf("\nStack is full ");
         else      {          top++;              s[top]=data;
         }
}
char pop()
{        char data;          if(isempty()==1)                         printf("\nStack is
empty ");
         else
                                                                                                     {          data=s[top];
                                                                                                                top--;
                        return data;
                                                                                                     }



                        switch(str[i])

                                   case '+':           op2=pop();
                                                                  op1=pop();
                                                                  push(op1+op2);
}  void postfix_eval(char str[20]) {          int i,op1,op2,val;
         for(i=0;str[i]!='\0';i++)
         {
                     {




                                                       break;
                        case '-':  op2=pop();
                                                       op1=pop();
                                                       push(op1-op2);
                                                       break;
                        case '*':            op2=pop();
                                                       op1=pop();
```

```
         push(op1*op2);
                                                       break;
```

```
                        case '/':  op2=pop();
                                                       op1=pop();
         push(op1/op2);
                                                       break;
  default:printf("Enter value of %c ",str[i]);
                                              scanf("%d",&val);
                                              push(val);

                     }
         }
         printf("Ans =%d ",pop());
}
main()
{        char str[20];       printf("Enter postfix string ");
         scanf("%s",str);
         postfix_eval(str);
}
```

Slip 4_1: Read the 'n' numbers from user and sort using bubble sort

Solution :

```
#include <stdio.h>
 void main(){    int a[20],i,n,temp;
   printf("enter number of elements :");    scanf("%d",&n);    for(i=0;i<n;i++)
   {
      printf("enter number :");       scanf("%d",&a[i]);
   }
   printf("/n before sorting                                                   :");
for(i=0;i<n;i++)
   {
      printf("%d\t",a[i]);
   }
   printf("/n after sorting                                                    :");
for(i=0;i<n;i++)
   {
      if(a[i]>a[i+1])
      {         temp=a[i];
a[i]=a[i+1];          a[i+1]=temp;
      }    }
   for(i=0;i<n;i++)
   {
      printf("%d\t",a[i]);
   }
}
```

**Slip 5_1,
Slip**

14_1: .Create a random array of n integers. Accept a value x from user and use linear search algorithm to check whether the number is present in the array or not and output the position if the number is present.

Solution :

```
#include<stdio.h>
 void linearsearch(int a[10],int n,int sr)
{        int i,p,cnt=0;
         for(i=0;i<n;i++)
         {
                  if(a[i]==sr)              {         p=i; //store position
                           cnt++;                     break;
                  }
         }
         if(cnt>=1)
                  printf("element found at %d position",p);
         else
                  printf("element NOT found ");
} main()
{
            int n,i,sr,a[10];
            printf("enter how many values");
            scanf("%d",&n);
            for(i=0;i<n;i++)
            {
                     printf("enter values");
                     scanf("%d",&a[i]);
            }
            printf("\n enter search element");
            scanf("%d",&sr);
            linearsearch(a,n,sr);
}
```

Slip 5_2 ,Slip 11_2 ,Slip 23_1 : Implement a priority queue library (PriorityQ.h) of integers using a static implementation of the queue and implement the below two operations. 1) Add an element with its priority into the queue. 2) Delete an element from queue according to its priority.

Solution :

Header File : PriorityQ.h

```
#include<stdio.h> int Q[20]; int f,R; void
init()
{
         f=R=-1;
}
```

```
int
isempty()
```

```
{
         if(f==R)                      return 1;          else
                  return 0;
}  int isfull()
{
         if(R==19)
                  return 1;
         else return 0;
}
void Add(int no)
{        int i;    if(isfull()==1)
                  printf("Queue is Full ");
         else
         {
                  for(i=R;i>f;i--)
                  {         if(no<Q[i])
                                                        Q[i+1]=Q[i];
                                                 else
                                                        break;

                  }
                  Q[i+1]=no;
                  R++;
         }
}
int Delete()
{        int no;  if(isempty()==1)
                  printf("Queue is
empty ");
         else
         {                  f++;
                  no=Q[f];
         }
         return no;
} void display()
{
         int i;
         for(i=f+1;i<=R;i++)
         {        printf("%d ",Q[i]);
         }
}
```

Program File :

```
#include<stdio.h> #include "PriorityQ.h"
main()
{
           int n,ch;
         init();
         do
         {
 printf("\n\n1.Add \n2.Delete \n3.Display \n0.EXit");
```

```
         printf("\nEnter choice ");
                  scanf("%d",&ch);
```

```
                  switch(ch)
                  {         case 1:printf("\nEnter element ");
                                    scanf("%d",&n);
                                    Add(n);
         break;
                            case 2:if(isempty()==1)
                                    printf("\nQueue is empty ");
                                    else
                                    printf("deleted elemet =%d
",Delete());
                                    break;
                            case 3:display();
                                    break;                  case 0:break;
                                 default:printf("\nInvalid choice ");
                       }
             } while(ch!=0);
}
```

Slip 6_1, Slip 15_1, Slip 18_1, Slip 19_1 : Sort a random array of n integers (accept the value of n from user) in ascending order by using selection sort algorithm.

Solution :

```
#include<stdio.h>

main()
{        int i,a[10],n,min,pos,j,temp;
         printf("Enter how many elements ");
         scanf("%d",&n);
         for(i=0;i<n;i++)
         {
                  a[i]=rand()%100;
         }
         printf("\nBefore array sorting ");            for(i=0;i<n;i++)              {
         printf("%d ",a[i]);
         }
         for(j=0;j<n-1;j++)
         {

                  min=a[j];                     pos=j;
         for(i=j+1;i<n;i++)                     {         if(a[i]<=min)
                           {         min=a[i];
                  pos=i;
                           }
                  }
                  temp=a[j];                     a[j]=min;
                  a[pos]=temp;
         }
```

```
         printf("\nSorted array is ");          for(i=0;i<n;i++)
         {        printf("%d ",a[i]);
```

```
         }
}
```

Slip 6_2 : Implement a queue library (dyqueue.h) of integers using a dynamic (linked list) implementation of the queue and implement init, enqueue, dequeue, isempty, peek operations.

Solution :

Header File : dyqueue.h

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

struct node
{
              int data;
              struct node *next;
};
struct node *f,*r;
 void init()
{
         f=NULL;           r=NULL;
}
int isempty()
{
         if(f==NULL)        return 1;
         else
           return 0;
}
void enqueue()
{
         struct node*nw;              int n;
         nw=(struct node*)malloc(sizeof(struct node));       nw->data=n;       nw->next=NULL;
             if(f==NULL)
         {                    f=nw;               r=nw;
         }
         else
         {
                  r->next=nw;
                  r=r->next;
         }
}
int dequeue()
{
         int n;
         struct node *temp;          if(isempty()==1)
         printf("queue is empty");
         else
         {
```

```
         temp=f;                       f=f->next;                      n=temp-
>data;
```

```
                  free(temp);
         }
} int peek()
{
         return f->data;
}
```

Program File :

```
#include<stdio.h>
#include"dyqueue.h"
 main()
{
         int ch,no;
         init();    do
         {
                  printf("\n1.enqueue \n2.dequeue \n3.peek \n0.exit");
                  printf("enter choice");
                  scanf("%d",&ch);
                  switch(ch)
                  {
                           case 1:printf("enter data:");                                          scanf("%d",&no);
                                     enqueue(no);                                     break;
                           case 2:if(isempty()==1)                              printf("\n queue is empty");
                                     else
     printf("dequeue element=%d",dequeue());
                                     break;
                     case 3:printf("top element =%d",peek());
                                     break;
                           case 0:break;
                  }
         }while(ch!=0);
}
```

Slip 7_1 : Sort a random array of n integers (accept the value of n from user) in ascending order by using quick sort algorithm.

Solution :

```
#include<stdio.h>
void quicksort(int a[10],int lb,int ub);

main()
{        int n ,a[10],i,sr,j,temp;
```

```
            printf("\n enter no of elements");
            scanf("%d",&n);
            for(i=0;i<n;i++)
            {
                     //printf("Enter no ");
                     a[i]=rand()%100;
            }
            printf("\n Before sorted array is ");
            for(i=0;i<n;i++)
            {
                     printf("%d ",a[i]);
            }
            quicksort(a,0,n-1);
            printf("\n Sorted array is ");
            for(i=0;i<n;i++)
         {
                  printf("%d ",a[i]);
         }
 }
void quicksort(int a[10],int lb,int ub)
{        int key,temp,i,j;
         if(lb<ub)
         {
                  i=lb+1;            key=a[lb];                  j=ub;

                  while(i<=j)
                  {

                  while(a[i]<=key && i<=ub)
                           i++;
                  while(a[j]>key && j>=lb)
                           j--;
                  if(i<j)
                  {
                           temp=a[i];                          a[i]=a[j];
                  a[j]=temp;
                  }
                  }
                  //swap key and a[j]                temp=a[j];
         a[j]=a[lb];              a[lb]=temp;
                  quicksort(a,lb,j-1);                quicksort(a,j+1,ub);
         }
}
```

Slip7_2 : Write a program that checks whether a string of characters is palindrome or not. The function should use a stack library (cststack.h) of stack of characters using a static implementation of the stack

Solution :

Header File : cststack.h

```
#include<stdio.h>
char s[20];
int top;

void init()
{
         top=-1;
}
int isempty()
{        if(top==-1)
                  return 1;
         else return 0;
}
int isfull()
{        if(top==19)
                  return 1;           else
                  return 0;
}
void push(char data)
{
         if(isfull()==1)                  printf("\nStack is full ");
         else      {           top++;
                        s[top]=data;
              }
}  char pop()
{        char data;         if(isempty()==1)                          printf("\nStack is
empty ");
         else
         {         data=s[top];
                  top--;
                  return data;
         }
}  int peek()
{        return s[top];
}
```

Program File :

```
#include<stdio.h>
#include"cststack.h"
 void main()
{
            char str[20];
            int count=0,i;
            char ch;
            printf("enter string");
            scanf("%s",str);
            init();
            for(i=0;i<=str[i]!='\0';i++)
            {
                     push(str[i]);
         }
         for (i=0;i<=strlen(str)/2;i++)
         {
                  ch = pop();                  if(ch!=str[i])
                  {
                           count++;                              break;
                  }
         }
                           if(count==0)
                           {
                           printf("The string is palindrome");
                           }
                           else
                           {
   printf("The string is not palindrome");
                           }

}
```

Slip 8-1: Implement a list library (singlylist.h) for a singly linked list of integer With the operations create, delete specific element and display. Write a menu driven program to call these operations.

Solution :

Header File : singlylist.h

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

```
void create()
{           int n,i;
            struct node *s;
            printf("Enter how many nodes ");
            scanf("%d",&n);

            f=          ( struct node *)malloc(sizeof(struct node));
            printf("Enter data ");
            scanf("%d",&f->data);
            s=f;
            for(i=1;i<n;i++)
            {
                        s->next=(struct node*)malloc(sizeof(struct node));
                        s=s->next;
                        printf("Enter data ");
                  scanf("%d",&s->data);
        }
        s->next=NULL;
}
void display()
{       struct node *s;
        for(s=f;s!=NULL;s=s->next)
        {
                printf("| %d |-> ",s->data);
        }
}
void Delete()
{       int p,cnt=0,i;      struct node *temp,*s;
        printf("Enter position to delete a node ");
        scanf("%d",&p);            for(s=f;s!=NULL;s=s->next)
        {          cnt++;
        }
        if(p==1)
        {
                temp=f;                         f=f->next;
        free(temp);
        }
        else if(p==cnt)
        {
                for(i=1,s=f;i<p-1;i++)
                {          s=s->next;
                }
                temp=s->next;               s->next=NULL;
                free(temp);
        }
        else if(p>1 && p<cnt)
        {
                for(i=1,s=f;i<p-1;i++)
                {          s=s->next;
                }
                temp=s->next;               s->next=temp->next;
                free(temp);
        }
```

```
        else
}

                    printf("Invalid Position ");
```

Program File :

```
#include<stdio.h>
                                                            #include"singlylist.h"


main()
{       int ch;
        do
        {
        printf("\n1.Create\n2.Display\n3.Delete \n0.Exit");
        printf("Enter choice ");  scanf("%d",&ch);
        switch(ch)
        {
                case 1:create();
                                break;
                case 2:display();
                                break;
                case 3:Delete();
                                break;
                case 0:break;
                default:printf("\nInvalid choice");

        }
        }while(ch!=0);
}
```

Slip 9_1,Slip 25_2 : Write a program to convert an infix expression of the form (a*(b+c)*((da)/b)) into its equivalent postfix notation. Consider usual precedence's of operators. Use stack library of stack of characters using static implementation.

Solution :

Header File : stack.h

```
#include<stdio.h> char s[20]; int top;  void
init()
{
        top=-1;
}
int isempty()
{       if(top==-1)
```

```
                return 1;
        else return 0;
}
int isfull()
{       if(top==19)
                return 1;
        else
        return 0;
}
void push(char data)
{
        if(isfull()==1)              printf("\nStack is full ");
        else      {         top++;             s[top]=data;
        }
}  char pop()
{       char data;       if(isempty()==1)                        printf("\nStack is
empty ");
        else
        {           data=s[top];
                top--;
                return data;
        }
}  int peek()
{       return s[top];
}
```

Program File :

```
#include<stdio.h>
#include "stack.h"
 int priority(char ch)
{       switch(ch)
        {
                case '(':return 0;                   case '+':
                case '-':return 1;                   case '*':
                case '/':return 2;                   case '^':
                case '$':return 3;
        }
        return 0;
}  void convert(char str[20])
{
        int i,j=0;          char post[20],ch,ch1;     init();
                                                              for(i=0;str[i]!='\0';i++)
        {          ch=str[i];
                switch(ch)
                {

                        case '(':push(ch);
                                            break;
                        case '+':
                        case '-':
                        case '*':
                        case '/':
                        case '$':
```

```
                                                                              case
'^':
                while(!isempty() && (priority(peek())>=priority(ch)))
                        {
                                post[j]= pop();
                                j++;
                        }
                        push(ch);                                    break;
                case ')':while((ch1=pop())!='(')
                                {          post[j]=ch1;
                                           j++;
                                }
                                break;
                default:post[j]=ch;
                                j++;

                }
        }
        while(!isempty())           {          post[j]=pop();
                j++;     }          post[j]='\0';
        printf("\n Postfix string = %s ",post);
}
main()
{
        char infix[20];
        printf("\nEnter the infix expression ");
        scanf("%s",infix);
        convert(infix);
}
```

Slip 9_1: Read the data from the 'employee.txt' file and sort on age using Counting sort or Quick sort and write the sorted data to another file 'sortedemponage.txt'.

Slip 28_2 : Read the data from the 'employee.txt' file and sort on age using Merge sort or Quick sort and write the sorted data to another file 'sortedemponage.txt

Solution : Using Quick sort

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct employee
{
            char name[20];
            int age;
} emp[10];

int readFile(struct employee a[])
{
```

```
        int i=0;  FILE *fp;
        if((fp=fopen("emp.txt","r"))!=NULL)
        {
                while(!feof(fp))
                {
                        fscanf(fp,"%s%d",&a[i].name,&a[i].age);
                        i++;
                }
        }
        return i-1;
}
void quicksort(struct employee a[10],int lb,int ub)
{       int i,j;   struct employee key,temp;          if(lb<ub)
        {
                i=lb+1;              key=a[lb];                     j=ub;

                while(i<=j)
                {


                while(a[i].age<=key.age && i<=ub)
                        i++;
                while(a[j].age>key.age && j>=lb)
                        j--;
                if(i<j)
                {
                          temp=a[i];                            a[i]=a[j];
                                          a[j]=temp;
                }
                }
                //sway key and a[j]                    temp=a[j];
                   a[j]=a[lb];
                   a[lb]=temp;
                   quicksort(a,lb,j-1);
                   quicksort(a,j+1,ub);
            }
}
void writeFile(struct employee a[],int n)
{
            int i=0;
            FILE *fp;
            if((fp=fopen("sortedemp_quick_age.txt","w"))!=NULL)
            {
                     for(i=0;i<n;i++)
                     {          fprintf(fp,"%s %d\n",a[i].name,a[i].age);
                     }
            }
}
main()
{       int n;    n=readFile(emp);          if(n==-1)                     printf("File
not found ");
        else
        {

        quicksort(emp,0,n-1);     writeFile(emp,n);
```

```
        printf("File Sorted ");
        }
}
```

Slip 10_1 ,22_1: Implement a linear queue library (st_queue.h) of integers using a static implementation of the queue and implementing the init(Q), add(Q) and peek(Q) operations. Write a program that includes queue library and calls different queue operations

Solution :

Header File :st_queue.h

```
#include<stdio.h> int Q[20]; int f,R;
void init()
{       f=R=-1;
}
int isempty()
{
        if(f==R)
        return 1;
        else
        return 0;
}
int isfull()
{
        if(R==19)
                return 1;
        else return 0;
}
void Add(int no)
{
        if(isfull()==1)              printf("Queue is Full ");
        else
        {
                R++;
                Q[R]=no;
        }
}  int Delete()
{       int no;  if(isempty()==1)                    printf("Queue is empty ");
        else
        {               f++;
                no=Q[f];
        }
                return no;
}
void display()
{
```

```
                                                                       int
i;
        for(i=f+1;i<=R;i++)
        {          printf("%d ",Q[i]);
        }
}
```

Program File :

```
#include<stdio.h> #include "st_queue.h" main()
{       int n,ch;           init();
        do
        {
                printf("\n\n1.Add \n2.Delete \n3.Display \n0.EXit");
                printf("\nEnter choice ");
                scanf("%d",&ch);
                switch(ch)
                {          case 1:printf("\nEnter element ");
                                        scanf("%d",&n);
                                        Add(n);
        break;
                        case 2:if(isempty()==1)                         printf("\nQueue is empty ");
                                        else
                                        printf("deleted elemet =%d
",Delete());
                                        break;
                        case 3:display();
                                        break;
                        case 0:break;
                        default:printf("\nInvalid choice ");
                }
        }while(ch!=0);
}
```

Slip10_2, 30_1 : Read the data from the file "employee.txt" and sort on names in alphabetical order (use strcmp) using bubble sort or selection sort.

Solution :Using Bubble sort

```
#include<stdio.h>
#include<stdlib.h> #include<string.h> struct employee
{
        char name[20];   int age;
}emp[10];

int readfile(struct employee a[10])
{
        int i=0;  FILE*fp;
        if((fp=fopen("empl.txt","r"))!=NULL)
        {
```

```
        while(!feof(fp))
                {
  fscanf(fp,"%s%d",a[i].name ,&a[i].age);
                                i++;
                }
        }
        return i-1;
}
void writefile(struct employee a[],int n)
{
        int i;      FILE *fp;
        if((fp=fopen("bsort.txt","w"))!=NULL)
        {
                for(i=0;i<n;i++)
                       {
                       fprintf(fp,"%s %d \n",a[i].name ,a[i].age);
                       }
           }
}
void bubblesort(struct employee a[],int n)
{
           int i,j;
           struct employee temp;
           for(i=0;i<n-1;i++)
           {
           for(j=0;j<n-1;j++)
                       {
                                  if(strcmp(a[j].name,a[j+1].name)>0)
                                  {
                                    temp=a[j];                                  a[j]=a[j+1];
                              a[j+1]=temp;
                              }
                   }
        }
} main()
{
        int n;
        n=readfile(emp);            if(n==-1)
        printf("File is not found");
        else
        {
                bubblesort(emp,n);                    writefile(emp,n);
                printf("File is  found");
        }

}
```

Slip 11_1: Accept n values in array from user. Accept a value x from user and use sentinel linear search algorithm to check whether the number is present in the array or not and output the position if the number is present

Solution :

```
#include<stdio.h>
 void sentinelsearch(int a[10],int n,int sr)
{        int i,cnt=0;        a[n]=sr;            while(sr!=a[i])
         {         i++;
         }
         if(i<n)
                   printf("Element is found at %d position ",i);
             else
                        printf("element is not found ");

}
main()
{
             int n,i,sr,a[10];
             printf("enter how many values");
             scanf("%d",&n);
             for(i=0;i<n;i++)
             {
                        printf("enter values");
                        scanf("%d",&a[i]);
             }
             printf("\n enter search element");
         scanf("%d",&sr);
         sentinelsearch(a,n,sr); }
```

Slip 12_1 : Read the data from file 'cities.txt' containing names of cities and their STD codes. Accept a name of the city from user and use linear search algorithm to check whether the name is present in the file and output the STD code, otherwise output “city not in the list”.

Solution :

```
#include<stdio.h>

#include<stdlib.h>
#include<string.h>
struct city
{
         char name[20];
         int code;
}ct[10];  int readFile(struct city a[])
{
         int i=0;  FILE *fp;
         if((fp=fopen("city.txt","r"))!=NULL)
         {
                   while(!feof(fp))
                   {
                             fscanf(fp,"%s%d",&a[i].name,&a[i].code);
                             i++;
                                                                                                            }
         }
```

```
         return i-1;
}
void linearsearch(struct city a[10],int n,char sr[20])
{        int i,p,cnt=0;
                                                                                                    for(i=0;i<n;i++)
         {

                             if(strcmp(a[i].name,sr)==0)
                                       p=i;        //store position
                   {                   cnt++;
                                       break;

                   }
         }
                             printf("city is found and code is %d ",a[p].code);
         if(cnt>=1)
                             printf("city NOT found ");
         else

}
main()
{        int n;    char sr[20];       n=readFile(ct);
         if(n==-1)
                   printf("File not found ");
         else
         {
         printf("Enter city name to search ");
         scanf("%s",sr);
         linearsearch(ct,n,sr);

         }
}
```

Slip 12_2, Slip 15_2,Slip24_1 : Implement a circular queue library (cir_queue.h) of integers using a dynamic (circular linked list) implementation of the queue and implementing init(Q), AddQueue(Q) and DeleteQueue(Q) , peek(Q) operations. Write a menu driven program that includes queue library and calls different queue operations.

Solution :

Header File : cir_queue.h

```
#include<stdio.h> #include<stdlib.h> struct node
{
         int data;
         struct node *next;
}; struct node *r; void init()
{
```

```
              r=NULL;
}
int isempty()
{
              if(r==NULL)
                        return 1;
              else
                        return 0;
}
void Add(int n)
{
              struct node *nw;
              nw=(struct node *)malloc(sizeof(struct node));
              nw->data=n;
              if(r==NULL)
         {                   r=nw;
                   r->next=r;
         }
         else
         {
                   nw->next=r->next;                   r->next=nw;
                   r=r->next;
         }
}  int Delete()
{
         int no;
         struct node *temp;         temp=r->next;   if(r==temp->next)
         {         r=NULL;
         }
         else
         {
                   r->next=temp->next;
         }
         no=temp->data;             free(temp);
         return (no);
} int peek()
{
         return r->next->data;
}
```

Program File:

```
#include<stdio.h> #include "cir_queue.h" main()
{        int ch,no;           init();
         do
         {
                   printf("\n1.Add \n2.Delete \n0.Exit");
                   printf("Enter choice");
                   scanf("%d",&ch);
                   switch(ch)
                   {
                             case 1:printf("\n Enter element");
                                                 scanf("%d",&no);
```

```
         Add(no);                                              break;
                             case 2:if(isempty()==1)                                      printf("\n Queue is empty");
                                                 else
                                                           {
                                                                     printf("deleted element is %d",Delete());
                                                                     break;

                                                           }
                             case 0:break;
                             case 4:printf("Elemenent at peek %d ",peek());
                                                 break;
  default:printf("Invalid choice");
                   }
         }while(ch!=0);

}
```

Slip 13_1, Slip 20_1 , Slip 26_1, Slip 29_1,Slip28_1 : Implement a stack library (ststack.h) of integers using a static implementation of the stack and implementing the operations like init(S), S=push(S), isFull(S). Write a driver program that includes stack library and calls different stack operations.

Solution :

Header File : sstack.h

```
#include<stdio.h> char s[20]; int top;  void
init()
{
         top=-1;
}
int isempty()
{        if(top==-1)
                   return 1;
         else return 0;
}
int isfull()
{        if(top==19)
                        return 1;
              else
                        return 0;
                                                                      }
void push(char data)
                                                                      {
              if(isfull()==1)
                        printf("\nStack is full ");
              else
              {         top++;
                        s[top]=data;
              }
}
char                                                                                pop()
{        char data;         if(isempty()==1)                 printf("\nStack is
empty ");        else
```

```
         {         data=s[top];
                   top--;
                   return data;
         }
}
int peek()
{        return s[top];
}
```

Program File :

```
#include<stdio.h>
#include<stdlib.h> #include"sstack.h" main()
{
         int n,i=0,ch;       init();
         do
         {
         printf("\n1.push \n2.pop \n3.chech stack is empty or not
\n4.chech stack is full or not \n5.Peek \n0.exit");  printf("\neneter your choice ");
         scanf("%d",&ch);

         switch(ch)
         {
                   case 1: printf("enter elements");
                             scanf("%d",&n);                           push(n);
                             break;

                   case 2:printf("\ndeleted elements :%d",pop());
                             break;

                   case 3:if(isempty()==1)
                             printf("stack is empty");
                             else
                             printf("stack is not empty");
                             break;

                   case 4:if(isfull()==1)
                             printf("stack is full");
                             else
                             printf("stack is not full");
                             break;

                   case 5:printf("\ntop of elements:%d",peek());
                             break;

                   case 0: break;

         }
         }while(ch!=0);
}
```

Slip 16_1: Sort a random array of n integers (accept the value of n from user) in ascending order by using Counting sort algorithm

Solution :

```
#include<stdio.h>
void countingsort(int a[20],int n,int k)
{
         int count[50],b[30],i;       for(i=0;i<=k;i++)
         {         count[i]=0;
         }
         for(i=0;i<n;i++)
         {         ++count[a[i]];
         }
         for(i=1;i<=k;i++)
         {         count[i]=count[i]+count[i-1];
         }
         for(i=n-1;i>=0;i--)          {          b[--count[a[i]]]=a[i];
         }
         //copy sorted array b to original array b
         for(i=0;i<n;i++)
         {         a[i]=b[i];
         }

}

main()
{
         int a[20],n,i,max;            printf("Enter how many elements ");
         scanf("%d",&n);

            for(i=0;i<n;i++)
            {             a[i]=rand()%10;
            }
            printf("\n Before sort array is ");
            for(i=0;i<n;i++)
            {
                          printf("%d ",a[i]);
            }
            max=a[0];
            for(i=1;i<n;i++)
            {             if(a[i]>max)
                                    max=a[i];
            }
            countingsort(a,n,max);
         printf("\n Afer sorting array is ");
         for(i=0;i<n;i++)
         {         printf("%d ",a[i]);
         }
}
```

Slip 17_1 : 1 Implement a list library (singlylist.h) for a singly linked list. Create a linked list, reverse it and display reversed linked list

Solution :

Header File : singlylist.h

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
         int data;
         struct node *next;
}; struct node *f; void create()
{
         struct node *s;
         int n,i;
         printf("Enter how many nodes");            scanf("%d",&n);
         f=(struct node *)malloc(sizeof(struct node));
         printf("Enter data");      scanf("%d",&f->data);
         s=f;
         for(i=1;i<n;i++)
         {
  s->next=(struct node *)malloc(sizeof(struct node));
                   s=s->next;
                   printf("Enter data");                 scanf("%d",&s->data);
            }
            s->next=NULL;
                                                                      }


void display()
                                                                      {
            struct node *s;
            for(s=f;s!=NULL;s=s->next)
            {
                      printf("%d ->",s->data);
            }

}
void reverse()
{
         int cnt=0,i;         struct node *s;
         for(s=f;s!=NULL;s=s->next)
         {
                   cnt++;
         }
         while(cnt>0)
         {
```

```
         for(i=1,s=f;i<cnt;i++)
                   {
                             s=s->next;
                   }
                   printf("%d ->",s->data);
                   cnt--;
         }
}
```

Program File

```
#include<stdio.h>
#include"singlylist.h" main()
{
         create();            display();          reverse();
}
```

Slip 18_2 : Write a program that multiply two single variable polynomials. Each polynomial should be represented as a list with linked list implementation

Solution :

```
#include<stdio.h>
#include<stdlib.h>
 struct node
{        int coeff,exp;
         struct node *next;
};

struct node* create(struct                                           node *f)
 {
         int i,n;
   struct node *s;
printf("\nEnter no of
terms ");    scanf("%d",&n);
   printf("Enter term in descending order of power ");     f=(struct node *)malloc(sizeof(struct
node));     printf("\n Enter coeff ");     scanf("%d",&f->coeff);     printf("\n Enter power ");
scanf("%d",&f->exp);     s=f;
   for(i=1;i<n;i++)
         {
  s->next=(struct node *)malloc(sizeof(struct node));
                   s=s->next;
                   printf("\n Enter coeff ");
    scanf("%d",&s->coeff);     printf("\n Enter power ");
    scanf("%d",&s->exp);
         }
         s->next=NULL;
```

```
         return f;
}
```

```
void display(struct node *f) {   struct node *s;     for(s=f;s!=NULL;s=s-
>next)          {          printf("%dx^%d ->",s->coeff,s->exp);
        }
} int length(struct node *p)
{         int len=0;          struct node *s;    for(s=p;s!=NULL;s=s-
>next)
        {          len++;
        }
        return len;
}
struct node* Mult(struct node *p1,struct node *p2)
{
        struct node *t1,*t2,*t3=NULL,*nw;
        struct node *p3;


        for(t1=p1;t1!=NULL;t1=t1->next)
        {
                for(t2=p2;t2!=NULL;t2=t2->next)
                {
                        nw=(struct node*)malloc(sizeof(struct node));
                                                                              nw->next=NULL;
                              nw->coeff=t1->coeff*t2->coeff;
                              nw->exp=t1->exp+t2->exp;
                              if(t3==NULL)
                              {          p3=nw;
                                         t3=nw;
                              }
                              else
                              {          t3->next=nw;
                                         t3=t3->next;
                              }

                }
        }

        return p3;
} main()
{       struct node *p1=NULL,*p2=NULL,*p3=NULL;
        p1=create(p1);              p2=create(p2);

        printf("\n 1st Polynomial is : ");            display(p1);
        printf("\n 2nd Polynomial is : ");
        display(p2);       p3=Mult(p1,p2);
        printf("\n Multiplication of 2 Polynomial is ");      display(p3);

}
```

Slip 20_2, Slip 29_2 : There are lists where new elements are always appended at the end of the list. The list can be implemented as a circular list with the external pointer pointing to the last element of the list. Implement singly linked circular list of integers with append and display operations. The operation append(L, n), appends to the end of the list, n integers accepted from user.

Solution :

```
#include<stdio.h> #include<stdlib.h> struct node
{
        int data;
        struct node *next,*prev;
}; struct node *f; void create()
{
        struct node *s;    int i,n;
            printf("enter how many nodes");
            scanf("%d",&n);
            f=(struct node*)malloc(sizeof(struct node));
            printf("enter data");
            scanf("%d",&f->data);
            s=f;
            for(i=1;i<n;i++)
            {
                        s->next=(struct node*)malloc(sizeof(struct node));
                        s=s->next;
                        printf("enter data");
                        scanf("%d",&s->data);
            }
            s->next=f;
}

void display()
{
        struct node *s;
        printf("\nCircular linked list is::");
        s=f;      do
        {
                printf("%d->",s->data);                    s=s->next;
        }
        while(s!=f);
}
void append()
{
        struct node *nw,*s;
        int n,i;
        printf("\nenter how many new nodes");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
        nw=(struct node*)malloc(sizeof(struct node));      printf("\nenter new node of data");
        scanf("%d",&nw->data);

        s=f;
        do
        {                                                                              s=s-
>next;

        }while(s->next!=f);
```

```
         s->next=nw;      nw->next=f;
}
}  main()
{
         create();          display();          append();
         display();
}
```

Slip 21_2, Slip 24_2 : Read the data from the file "employee.txt" and sort on names in alphabetical order (use strcmp) using insertion sort or selection sort.

Solution : Using Insertion sort

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

struct employee {    char name[20];    int age;
}emp[10];
int readfile(struct employee a[])
{    int i=0;    FILE*fp;
   if((fp=fopen("emp.txt","r"))!=NULL)
   {
      while(!feof(fp))
      {
         fscanf(fp,"%s%d",&a[i].name,&a[i].age);          i++;
      }
   }
   return i-1;
}  void InsertionSort(struct employee a[],int n)
{  int i,j;     struct employee key;     for(i=1; i<n; i++)
   {       key=a[i];
      for(j=i-1; j>=0; j--)
      {
         if(strcmp(a[j].name,key.name)>0)
         {
            a[j+1]=a[j];
         }          else              break;
      }      a[j+1]=key;
   }
}
```

```
void writefile(struct employee a[],int n)
{
   int i=0;
   FILE*fp;
   if((fp=fopen("insertionsort.txt","w"))!=NULL)
   {
      for(i=0;i<n;i++)
      {
         fprintf(fp,"%s %d\n",a[i].name,a[i].age);
      }
   }
}

int main()
{    int n;    n=readfile(emp);    if(n==-1)
      printf("File Not Found");    else
   {
      InsertionSort(emp,n);       writefile(emp,n);       printf("File
Sorted");
   }
}
```

Slip 21_1: Write a program that reverses a string of characters. The function should use a stack library (cststack.h). Use a static implementation of the stack.

Solution

Header File : cststack.h

```
#include<stdio.h> char s[20]; int
top;  void init() {    top==-1;
}  int isempty() {    if(top==-1)
return 1;    else       return 0;
}  int isfull() {    if(top==19)                                         return
1;    else       return 0;
}  void push(char ch) {
if(isfull()==1)       printf("Stack is
full");    else    {       top++;
s[top]=ch;
   }
}  char pop() {    char ch;
if(isempty()==1)       printf("Stack is empty");    else    {
ch=s[top];       top--;       return ch;
   }
}
```

Program File :

```
#include<stdio.h> #include"stack.h" int
main() {    init();    char str[20];    int i;
```

```
   printf("Enter String: ");    scanf("%s",&str);    for(i=0;str[i]!='\0';i++)
   {
      push(str[i]);
   }
   printf("Reversed string: ");    while(!isempty())
   {
      printf("%c",pop());
   }
}
```

Slip 22_2: Read the data from file 'cities.txt' containing names of cities and their STD codes. Accept a name of the city from user and use sentinel linear search algorithm to check whether the name is present in the file and output the STD code, otherwise output "city not in the list".

Solution :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct city
{
             char name[20];
         int code;
}ct[10];
int readFile(struct city a[])
{
         int i=0;  FILE *fp;
         if((fp=fopen("city.txt","r"))!=NULL)
         {
                  while(!feof(fp))
                  {
  fscanf(fp,"%s%d",&a[i].name,&a[i].code);
                           i++;
                  }
         }
         return i-1;
} void sentinelsearch(struct city a[10],int n,int sr)
{        int i,cnt=0;
         a[n]=sr;
         while(strcmp(sr,a[i].name)!=0)
         {        i++;
         }
         if(i<n)
 printf("city is found and STD code is %d  ",a[i].code);
         else
                  printf("city is not found ");

}
main()
{        int n;    char sr[20];      n=readFile(ct);   if(n==-1)
         printf("File not found ");
         else
```

```
         {
         printf("Enter city name to search");
         scanf("%s",sr);             SentinelSearch(ct,n,sr);

         }
}
```

Slip 23_2: Read the data from file 'sortedcities.txt' containing sorted names of cities and their STD codes. Accept a name of the city from user and use binary search algorithm to check whether the name is present in the file and output the STD code, otherwise output "city not in the list".

Solution :

```
#include<stdio.h>
#include<stdlib.h> #include<string.h> struct city
{
         char name[20];
         int code;
}ct[10];
int readFile(struct city a[])
{
         int i=0;  FILE *fp;
         if((fp=fopen("sortedfile.txt","r"))!=NULL)
         {
                  while(!feof(fp))
                  {
                           fscanf(fp,"%s%d",&a[i].name,&a[i].code);
                           i++;
                  }
         }
         return i-1;
}
int binarysearch(struct city a[10],int lb,int ub,char sr[20])
{
         int mid=0;        while(lb<=ub)
         {
                  mid=(lb+ub)/2;

                  if(strcmp(a[mid].name,sr)==0)
                           return mid;

                  else if(strcmp(sr,a[mid].name)<0)
                           ub=mid-1;                  else
                  lb=mid+1;
         }
         return -1;
} main()
```

```
{            int n,p;
             char sr[20];
             n=readFile(ct);
             if(n==-1)
                        printf("File not found ");
             else
             {
             printf("Enter city name to search ");
             scanf("%s",sr);
             p=binarysearch(ct,0,n,sr);
             if(p>=0)
                        printf("\nCity is found and code =%d ",ct[p].code);
             else
                        printf("\nCity not found ");
             }
}
```

Slip27_1 : Read the data from the file and sort on names in alphabetical order (use strcmp) using Merge sort and write the sorted data to another file 'sortedemponname.txt'

Solution :

```
#include<stdio.h>
#include<stdlib.h> #include<string.h> struct employee
{
        char name[20];  int age; }emp[10];  int
readFile(struct employee a[])
{
        int i=0;  FILE *fp;
        if((fp=fopen("emp.txt","r"))!=NULL)
        {
                while(!feof(fp))
                {
  fscanf(fp,"%s%d",&a[i].name,&a[i].age);
                        i++;
                }
        }
        return i-1;
}
mergesort(struct employee a[10],int lb,int ub)
{       int mid;          if(lb<ub)
        {
```

```
                        mid=(lb+ub)/2;
                        mergesort(a,lb,mid);
                        mergesort(a,mid+1,ub);
                        merge(a,lb,mid,ub);

            }
}

merge(struct employee a[10],int lb,int mid,int ub)
{
            struct employee b[20];
            int k,i,j;
            k=0;
            i=lb;
            j=mid+1;
        while(i<=mid && j<=ub)
        {
                //if(a[i]<=a[j])
                if(strcmp(a[i].name,a[j].name)<0)
                {         b[k]=a[i];
                          i++;                        k++;
        }
                else
                {         b[k]=a[j];
                          j++;                        k++;
                }
        }
        while(i<=mid)
        {
                b[k]=a[i];                            i++;
                k++;
        }
        while(j<=ub)
        {
                b[k]=a[j];
                j++;             k++;
        }
        for(i=lb,k=0;i<=ub;k++,i++)
        {
                a[i]=b[k];
        }
}
void writeFile(struct employee a[],int n)
{
        int i=0;  FILE *fp;
        if((fp=fopen("sortedemp_merge.txt","w"))!=NULL)
        {

                for(i=0;i<n;i++)
                {         fprintf(fp,"%s %d\n",a[i].name,a[i].age);
                }
        }
}
main()
```

```
{       int n;

                printf("File not found ");

        n=readFile(emp);
        if(n==-1)

        else
        {

        mergesort(emp,0,n-1);    writeFile(emp,n);
        printf("File Sorted ");
        }
}
```

Slip 27_2 : Write a program that adds two single variable polynomials. Each polynomial should be represented as a list with linked list implementation.

Solution :

```
#include<stdio.h>
#include<stdlib.h>
 struct node
{       int coeff,exp;
        struct node *next;
};

struct node* create(struct node *f)
 {
        int i,n;
   struct node *s;    printf("\nEnter no of terms ");
scanf("%d",&n);
   printf("Enter term in descending order of power ");    f=(struct node *)malloc(sizeof(struct
node));    printf("\n Enter coeff");    scanf("%d",&f->coeff);    printf("\n Enter power  ");
scanf("%d",&f->exp);    s=f;
   for(i=1;i<n;i++)
        {
  s->next=(struct node *)malloc(sizeof(struct node));
                                                                                  s=s-
>next;
```

```
        printf("\n Enter coeff");
    scanf("%d",&s->coeff);    printf("\n Enter power  ");
    scanf("%d",&s->exp);
        }
            s->next=NULL;
            return f;
}
void display(struct node *f)
{   struct node *s;
   for(s=f;s!=NULL;s=s->next)
            {           printf("%dx^%d ->",s->coeff,s->exp);
            }
}
struct node *Add(struct node *p1,struct node *p2)
{
            struct node *t1,*t2,*t3=NULL,*nw;
            struct node *p3;
            t1=p1;t2=p2;
            printf("\n%d  %d",t1->exp,t2->exp);
        while(t1!=NULL && t2!=NULL)
        {          nw=(struct node*)malloc(sizeof(struct node));
                   nw->next=NULL;
                   if(t1->exp > t2->exp)
                   {
                             nw->exp=t1->exp;                          nw->coeff=t1->coeff;
                             t1=t1->next;
                   }
                   else if(t2->exp > t1->exp)
                   {
                             nw->exp=t2->exp;                          nw->coeff=t2->coeff;
                             t2=t2->next;
                   }
                   else
                   {         nw->exp=t1->exp;                          nw->coeff=t1->coeff+t2->coeff;
                             t1=t1->next;                   t2=t2->next;
                   }
                   if(t3==NULL)
                   {         p3=nw;                         t3=nw;
                   }
                   else
                   {         t3->next=nw;
                             t3=t3->next;
                   }
        }
        while(t1!=NULL)
        {
                   nw=(struct node*)malloc(sizeof(struct node));
                   nw->next=NULL;                   nw->exp=t1->exp;
        nw->coeff=t1->coeff;
                   t1=t1->next;

                   t3->next=nw;;
                   t3=t3->next;
        }
        while(t2!=NULL)
```

```
            {
                        nw=(struct node*)malloc(sizeof(struct node));
                        nw->next=NULL;
                        nw->exp=t2->exp;
                        nw->coeff=t2->coeff;
                        t2=t2->next;

                        t3->next=nw;
                        t3=t3->next;
            }
            return p3;
}
main()
{           struct node *p1=NULL,*p2=NULL,*p3=NULL;
            p1=create(p1);
        p2=create(p2);

        printf("\n 1st Polynomial is : ");            display(p1);
        printf("\n 2nd Polynomial is : ");
        display(p2);      p3=Add(p1,p2);
        printf("\n Addition of 2 Polynomial is ");            display(p3);

}
```

Slip 30_2 : Write a program that merges two ordered linked lists into third new list. When two lists are merged the data in the resulting list are also ordered. The two original lists should be left unchanged. That is merged list should be new one. Use linked implementation.

Solution :

```
#include<stdio.h> struct node
{
        int data;
        struct node *next;
};
 struct node* create(); void display(struct node*);
  struct node* create()
{       int n,i;   struct node *s,*f;          printf("Enter how many nodes ");
        scanf("%d",&n);

        f=        (struct node *)malloc(sizeof(struct node));            printf("Enter data ");
        scanf("%d",&f->data);
            s=f;
            for(i=1;i<n;i++)
            {
                        s->next=(struct node*)malloc(sizeof(struct node));
                        s=s->next;
                        printf("Enter data ");
                        scanf("%d",&s->data);
            }
            s->next=NULL;
            return f;
                                                                                                  }
```

```
void display(struct node *f)
{       struct node *s;
        for(s=f;s!=NULL;s=s->next)
        {
                printf("| %d |->  ",s->data);
        }
}
struct node* merge(struct node *f1,struct node *f2)
{       struct node *s;
        for(s=f1;s->next!=NULL;s=s->next)
        {
        }
        s->next=f2;
        return f1;
} main()
{
        struct node *f1,*f2,*f3;   f1=create();      f2=create();
        printf("1st linked list ");          display(f1);      printf("\n2nd
linked list ");   display(f2);

        f3=merge(f1,f2)  ;
        printf("\nAfter merging LL is ");
        display(f3);
}
```