Lab 1

#include <stdio.h>

int main()  
{  
int size;  
printf("How many elements you have from which you want to determine largest element=");  
scanf("%d",&size);  
int a[size];  
int i;  
printf("Enter %d number of elements in the array:\n",size);  
for(i=0;i<size;i++)  
{  
scanf("%d",&a[i]);  
}

int Max=a[0],j=0;  
for(i=1;i<size;i++)  
{  
if(a[i]>Max)  
{  
Max=a[i];  
j=i;  
}  
}  
printf("The largest element is %d which is at index %d", Max,j);

return 0;  
}

#include <stdio.h>

int main()  
{  
int size;  
printf("How many elements you have from which you want to determine smallest element=");  
scanf("%d",&size);  
int a[size];  
int i;  
printf("Enter %d number of elements in the array:\n",size);  
for(i=0;i<size;i++)  
{  
scanf("%d",&a[i]);  
}  
  
int Min=a[0],j=0;  
for(i=1;i<size;i++)  
{  
if(a[i]<Min)  
{  
Min=a[i];  
j=i;  
}  
}  
printf("The smallest element is %d which is at index %d", Min,j);

return 0;  
}

#include <stdio.h>

int main()  
{  
int size,ele,ind;  
printf("Enter size of array: ");  
scanf("%d",&size);  
int a[size];  
int i;  
printf("Enter %d elements: \n",size);  
for(i=0;i<size;i++)  
{  
scanf("%d",&a[i]);  
}  
  
printf("Enter element to be searched: ");  
scanf("%d",&ele);  
  
for(i=0;i<size;i++)  
{  
if(a[i]==ele)  
{  
ind=i;  
break;  
}  
}  
printf("The element %d is found at index %d",ele,ind);

return 0;  
}

Lab 2

int binarySearch(int a[], int low, int high, int x)  
{  
while (low<=high) {  
int mid;  
mid=low+(high-low)/2;  
if (a[mid]==x)  
return mid;  
if (a[mid] < x)  
low=mid+1;  
else  
high=mid-1;  
}  
return -1;  
}

void main()  
{  
int a[]= {6,9,4,20,18};  
int n=sizeof(a)/sizeof(a[0]);  
int x=20;  
int y=binarySearch(a,0,n,x);  
if (y==-1)  
printf("Element is not present in array");  
else  
printf("Element is present at index %d",y);  
}

#include<stdio.h>

int BinSearch(int \*A, int low,int high, int x)  
{

int mid;  
while(low<=high)  
{  
mid=(low+high)/2;  
if(x<A[mid])  
return BinSearch(A, low, mid - 1, x);  
else if (x>A[mid])  
return BinSearch(A, mid+1, high, x);else  
return mid;  
}  
return -1;  
}

int main()  
{  
int size;  
printf("How many elements you have in which you want to determine the index of given element=");  
scanf("%d",&size);  
int a[size];  
printf("Enter %d number of elements in the array in ascending order:\n",size);  
for(int i=0;i<size;i++)  
{  
scanf("%d",&a[i]);  
}  
int item;  
printf("Enter the element whose index position you want to determine=");  
scanf("%d",&item);  
int low=0, high=size-1;  
int index=BinSearch(a,low,high,item);  
if(index==-1)  
printf("Element not found");  
else  
printf("Element found at index position %d",index);  
}

Lab 3

#include<stdio.h>  
void SelectionSort(int \*A, int n)  
{  
int temp;

for(int i=0;i<n-1;i++)  
{  
int min=i;  
for(int j=i+1;j<n;j++)  
if (A[j]<A[min])  
min=j;  
  
temp=A[i];  
A[i]=A[min];  
A[min]=temp;

}  
}

int main()  
{  
int size;  
printf("Enter the size of the array=");  
scanf("%d",&size);  
int a[size];  
printf("Enter %d number of elements in the array in random order:\n",size);  
for(int i=0;i<size;i++)  
{  
scanf("%d",&a[i]);  
}  
SelectionSort(a,size);  
printf("The sorted array is as follows:");  
for(int i=0;i<size;i++)  
{  
printf("\n%d",a[i]);  
}  
}

#include<stdio.h>  
void InsertionSort(int \*A, int n)  
{  
int i,k,j;  
for (i=1;i<n;i++){  
k=A[i];  
j=i-1;  
while (j>=0 && A[j]>k){  
A[j+1]=A[j];  
j=j-1;  
}  
A[j+1]=k;  
}  
}

int main()  
{  
int size;  
printf("Enter the size of the array=");  
scanf("%d",&size);  
int a[size];  
printf("Enter %d number of elements in the array in random order:\n",size);  
for(int i=0;i<size;i++)  
{  
scanf("%d",&a[i]);  
}  
InsertionSort(a,size);  
printf("The sorted array is as follows:");  
for(int i=0;i<size;i++)  
{  
printf("\n%d",a[i]);  
}  
}

Lab 4

#include<stdio.h>  
void Adjust(int \*a, int i, int n)  
{  
int j=2\*i, item=a[i];  
while(j<=n)  
{  
if((j<n) && (a[j]<a[j+1]))  
j=j+1;  
if(item>=a[j])  
break;  
a[j/2]=a[j];  
j=2\*j;  
}  
a[j/2]=item;  
}

void Heapify(int \*a,int n)  
{  
for(int i=n/2;i>=1;i--)  
{  
Adjust(a,i,n);  
}  
}

void HeapSort(int \*a,int n)  
{  
int t;  
Heapify(a,n);  
for(int i=n;i>=2;i--)  
{  
t=a[i];a[i]=a[1];a[1]=t;  
Adjust(a,1,i-1);  
}  
}

void main()  
{  
int size;  
printf("Enter the size of the array=");  
scanf("%d",&size);  
int a[size+1];  
printf("Enter %d number of elements in the array in random order:\n",size);  
for(int i=1;i<=size;i++)  
{  
scanf("%d",&a[i]);  
}  
HeapSort(a,size);  
printf("The sorted array is as follows:");  
for(int i=1;i<=size;i++)  
{  
printf("\n%d",a[i]);  
}  
}

Lab 5

#include<stdio.h>  
int a[100],b[100];  
void Merge(int low, int mid, int high)  
{  
int h=low, i=low, j=mid+1,k;  
while((h<=mid) && (j<=high))  
{  
if(a[h]<=a[j])  
{  
b[i]=a[h];h=h+1;  
}  
else  
{  
b[i]=a[j];j=j+1;  
}  
i=i+1;  
}  
if(h>mid)  
for(k=j;k<=high;k++)  
{  
b[i]=a[k];i=i+1;  
}  
else  
for(k=h;k<=mid;k++)  
{  
b[i]=a[k];i=i+1;  
}  
for(k=low;k<=high;k++)  
{  
a[k]=b[k];  
}  
}

void MergeSort(int low,int high)  
{  
int mid;  
if(low<high)  
mid=(low+high)/2;  
MergeSort(low,mid);  
MergeSort(mid+1,high);  
Merge(low,mid,high);  
}

void main()  
{  
int size;  
printf("Enter the size of the array (not more than 99) =");  
scanf("%d",&size);  
printf("Enter %d number of elements in the array in random order:\n",size);  
for(int i=1;i<=size;i++)  
{  
scanf("%d",&a[i]);  
}  
MergeSort(1,size);  
printf("The sorted array is as follows:");  
for(int i=1;i<=size;i++)  
{  
printf("\n%d",a[i]);  
}  
}

#include<stdio.h>  
int a[10];  
int partition(int l, int r)  
{  
int i=l,j=r,pivot=a[l],temp;  
do  
{  
do  
i=i+1;  
while((pivot>a[i]) && i<r);  
  
while((pivot<a[j]) && j>l)  
j=j-1;  
  
if(i<j)  
{temp=a[i];a[i]=a[j];a[j]=temp;}  
}while(i<j);  
a[l]=a[j];  
a[j]=pivot;  
return j;  
}

void quicksort(int start,int end)  
{  
int pos;  
if(start<end){  
pos=partition(start,end);  
quicksort(start,pos-1);  
quicksort(pos+1,end);  
}  
}  
int main()  
{  
int i;  
printf("Enter any 10 elements of the array\n");  
for(i=0;i<=9;i++)  
{  
scanf("%d",&a[i]);  
}  
quicksort(0,9);  
printf("array after sorting is:\n");  
for(i=0;i<10;i++)  
{  
printf("%d\n",a[i]);  
}  
return 0;  
}

LAB 6

#include <stdio.h>

void order(int \*p, int \*w, int n)

{

    double ratio[n];

    for(int i=0;i<n;i++)

    {

        ratio[i]=(double)p[i]/(double)w[i];

    }

    for(int j=0;j<n;j++)

    {

        for(int k=0;k<n-j-1;k++)

        {

            if (ratio[k]<ratio[k+1])

            {

                int t1,t2;

                double t3;

                t3=ratio[k];

                ratio[k]=ratio[k+1];

                ratio[k+1]=t3;

                t1=p[k];

                t2=w[k];

                p[k]=p[k+1];

                p[k+1]=t1;

                w[k]=w[k+1];

                w[k+1]=t2;

            }

        }

    }

}

void knapsack(int \*p, int \*w, double \*x, int m, int n)

{

    int U=m;

    int check=1;

    int temp;

    for(int i=0; i<n; i++)

    {

        if (w[i]>U)

        {

            check=2;

            temp=i;

            break;

        }

        x[i]=1.0;

        U=U-w[i];

    }

    if(check==2)

    {

        x[temp]=(double)U/(double)w[temp];

    }

}

int main()

{

    int n;

    int m;

    int p[]={25,24,15};

    int w[]={18,15,10};

    printf("Enter number of elements(n): ");

    scanf("%d",&n);

    printf("\nEnter size of knapsack(m): ");

    scanf("%d",&m);

    for(int i=0;i<n;i++)

    {

        //input elements for p,w

    }

    double x[n];

    double prof;

    for(int i=0;i<n;i++)

    {

        x[i]=0.0;

    }

    order(p,w,n);

    knapsack(p,w,x,m,n);

    for(int i=0;i<n;i++)

    {

        printf("%f\n",x[i]);

        prof+=p[i]\*x[i];

    }

    printf("profit=%f",prof);

    return 0;

}

LAB 7

#include <stdio.h>

#include <limits.h>

int minDistance(double \*dist, int\* s,int n)

{

int min=INT\_MAX;

int min\_index;

for(int l=0;l<n;l++)

{

if(s[l]==0 && dist[l]<=min)

{

min=dist[l];

min\_index=l;

}

}

return min\_index;

}

void ShortestPath(int v,int cost[8][8],double \*dist,int n)

{

int s[8];

for(int i=0;i<n;i++)

{

s[i]=0;

dist[i]=(double)cost[v][i];

}

s[v]=1;

dist[v]=0.0;

for(int num=1;num<n;num++)

{

int u;

u=minDistance(dist,s,n);

s[u]=1;

for(int m=0;m<n;m++)

{

if(cost[u][m]!=0 && cost[u][m]!=INT\_MAX && s[m]==0)

{

if(dist[m]>dist[u]+(double)cost[u][m])

{

dist[m]=dist[u]+(double)cost[u][m];

}

}

}

}

}

int main()

{

int v,n;

v=4;

n=8;

int cost[8][8]={{0,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX},

{300,0,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX},

{100,800,0,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX},

{INT\_MAX,INT\_MAX,1200,0,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX},

{INT\_MAX,INT\_MAX,INT\_MAX,1500,0,250,INT\_MAX,INT\_MAX},

{INT\_MAX,INT\_MAX,INT\_MAX,1000,INT\_MAX,0,900,1400},

{INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,0,1000},

{1700,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,INT\_MAX,0}};

double dist[8];

ShortestPath(v,cost,dist,n);

for(int i=0;i<n;i++)

{

printf("%f\n",dist[i]);

}

return 0;

}

LAB 12

#include <stdio.h>

void NextValue(int k,int G[5][5],int m,int x[5], int n)

{

while(1)

{

int i;

x[k]=(x[k]+1)%(m+1);

if(x[k]==0)

{

return;

}

for(i=0;i<n;i++)

{

if(G[k][i]!=0 && x[k]==x[i])

break;

}

if(i==n)

return;

}

}

int mcolouring(int k, int G[5][5], int m, int x[5], int n)

{

while(1)

{

NextValue(k,G,m,x,n);

if(x[k]==0)

return 0;

if(k==(n-1))

{

for(int i=0;i<n;i++)

{

printf("node %d = colour %d\n",i+1,x[i]);

}

return 1;

}

else

return(mcolouring(k+1,G,m,x,n));

}

}

void main()

{

int G[5][5]={{ 0, 1, 1, 0, 1},

{ 1, 0, 1, 0, 1},

{ 1, 1, 0, 1, 0},

{ 0, 0, 1, 0, 1},

{ 1, 1, 0, 1, 0}

};

int x[5]={0,0,0,0,0};

int m=3;

int n=5;

int k=0;

int p=mcolouring(k,G,m,x,n);

if(p==1)

{

printf("Complete");

}

else

printf("Solution not possible!");

}