# Practical 1

## Aim: Implement factorial algorithm using iterative and recursive manner. Code:

import time

n=int(input("Enter Last Number:\n"))

def iter(n):

fact=1

for i in range(1,n+1):

fact=fact\*i

def recursive(n):

if n==0 or n==1:

return 1

else:

return n\*recursive(n-1)

print(":::::::::::::Enter a Choice:::::::::::::")

x=int(input("1.For Iterative\n2.For Recursive:"))

if x==1:

a=time.time()

print(a)

iter(n)

b=time.time()

print(b)

print("Time Taken By Iterative Method:",b-a,"seconds")

elif x==2:

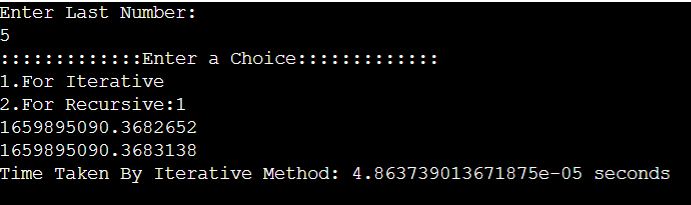
a=time.time()

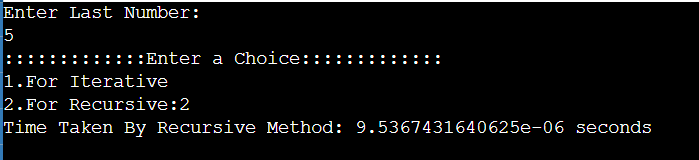
recursive(n)

b=time.time()

print("Time Taken By Recursive Method:",b-a,"seconds")

**Output:**

****



**Practical 2**

**Aim: Following is the data of height of 10 students of Sports class in school. Lined up in a random order in front of the teacher, who’s put to the task of lining all up in an ascending order of height. Now your task is to help your teacher in arranging them using following set of data and measure their execution time and time complexity.**

**Height:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Student** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **Height** | **89** | **42** | **100** | **93** | **11** | **234** | **30** | **82** | **22** | **75** |

**Code:**

from time import time

c1,c2,c3=0,0,0

n=int(input("Enter the number of elements:"))

stud=[]

height=[]

for i in range(1,n+1):

stud.append(i)

h=int(input("Enter height of student:"))

height.append(h)

def general():

c1,c2,c3=0,0,0

for i in range(1,n+1):

c1+=1

for j in range(1,n+1):

c2+=1

if height[j]>height[i]:

height1[i],height[j]=height[j],height[i]

stud[i],stud[j]=stud[j],stud[i]

c3+=1

print("Outer loop used:",c1,"times")

print("Inner loop used:",c2,"times")

print("If condition used:",c3,"times")

def best():

c1,c2,c3=0,0,0

for i in range(n):

height1.append(i)

for i in range(n):

c1+=1

for j in range(n):

c2+=1

if height1[j]>height1[i]:

height1[i],height1[j]=height1[j],height1[i]

stud[i],stud[j]=stud[j],stud[i]

c3+=1

print("Outer loop used:",c1,"times")

print("Inner loop used:",c2,"times")

print("If used:",c3,"times")

def worst():

c1,c2,c3=0,0,0

list2=[]

for i in range(n,0,-1):

list2.append(i)

for i in range(n):

c1+=1

for j in range(n):

c2+=1

if list2[j]>list2[i]:

list2[i],list2[j]=list2[j],list2[i]

stud[i],stud[j]=stud[j],stud[i]

c3+=1

print("Outer loop used:",c1,"times")

print("Inner loop used:",c2,"times")

print("If used:",c3,"times")

def average():

c1,c2,c3=0,0,0

for i in range(n):

height1.append(i)

random.shuffle(height1)

for i in range(n):

c1+=1

for j in range(n):

c2+=1

if height1[j]>height1[i]:

height1[i],height1[j]=height1[j],height1[i]

stud[i],stud[j]=stud[j],stud[i]

c3+=1

print("Outer loop used:",c1,"times")

print("Inner loop used:",c2,"times")

print("If used:",c3,"times")

print("Enter a Case:")

x=int(input("1.For Best Case\n2.For Worst Case\n3.For Average Case:\n4.For General Case:"))

if x==1 :

a=time()

best()

elif x==2 :

a=time()

worst()

elif x==3 :

a=time()

average()

elif x==4 :

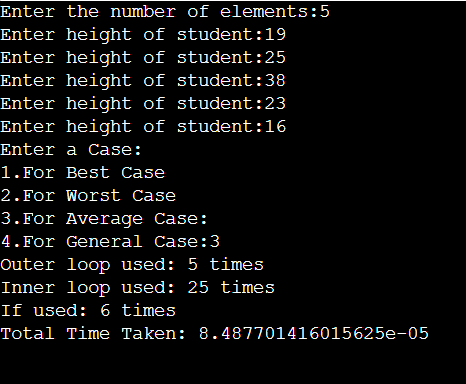
a=time()

general()

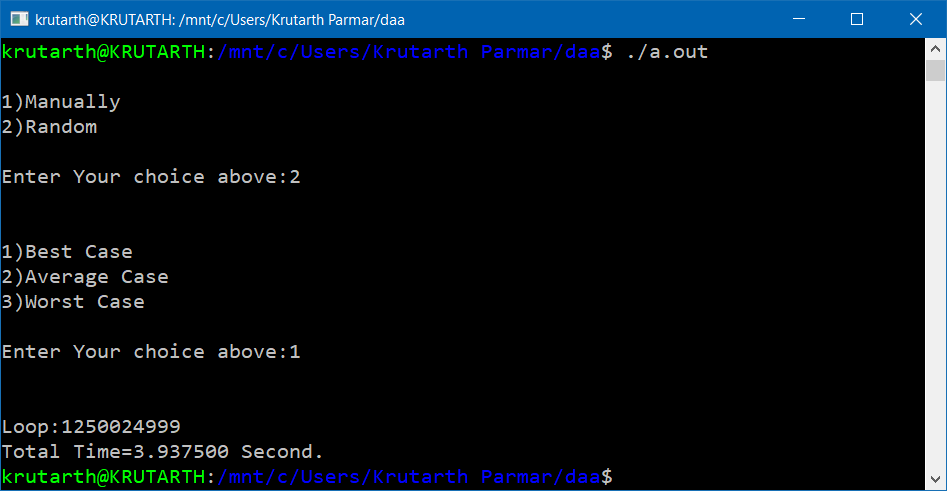
b=time()

print("Total Time Taken:",b-a)

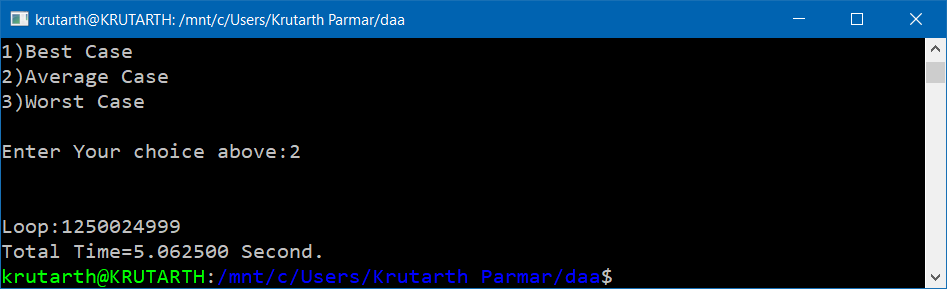
## Output:

****

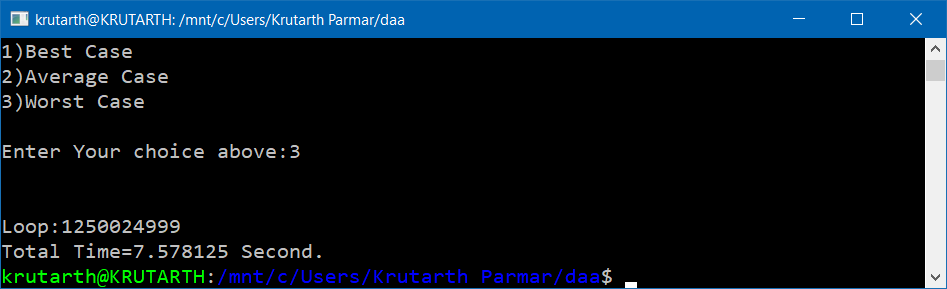
**Best Case:**



**Average Case:**



**Worst Case:**



**Practical 3**

**Aim: Implement insertion sort algorithm and perform its best case, average case and worst case analysis.**

**Code:**

#include<stdio.h> #include<time.h> clock\_t st,et;

int a[50000],n,i,j,k,key,c,cc,lc=0; double tt;

void insertion()

{

for(i=1;lc++,i<n;i++)

{

key=a[i]; j=i-1;

while(a[j]>key && j>=0)

{

a[j+1]=a[j]; j--;

}

a[j+1]=key;

}

}

void main()

{

printf("\n1)Manually"); printf("\n2)Random"); printf("\n\nEnter Your choice above:"); scanf("%d",&c);

if(c==1)

{

printf("Enter Numer of elemant:"); scanf("%d",&n);

printf("Enter elemant:"); for(i=0;i<n;i++) scanf("%d",&a[i]); st=clock(); for(i=1;lc++,i<n;i++)

{

key=a[i]; j=i-1;

while(a[j]>key && j>=0)

{

a[j+1]=a[j]; j--;

}

a[j+1]=key; printf("\nPass %d:",i+1); for(k=0;k<n;k++) printf("%d ",a[k]);

}

et=clock();

}

else if(c==2)

{

n=50000;

printf("\n\n1)Best Case"); printf("\n2)Average Case"); printf("\n3)Worst Case"); printf("\n\nEnter Your choice above:"); scanf("%d",&cc);

switch(cc)

{

case 1:

for(i=0;i<n;i++) a[i]=i; st=clock(); insertion(); et=clock(); break;

case 2:

for(i=0;i<n/2;i++) a[i]=i;

for(i=n/2,j=49999;i<n;i++,j--) a[i]=j;

st=clock(); insertion(); et=clock(); break;

case 3:

for(i=0,j=49999;i<n;i++,j--) a[i]=j;

st=clock(); insertion(); et=clock(); break;

}

}

else

default:

printf("Wrong Choice"); break;

printf("Wrong Choice");

tt=(double) (et-st)/CLOCKS\_PER\_SEC; if(c==1)

{

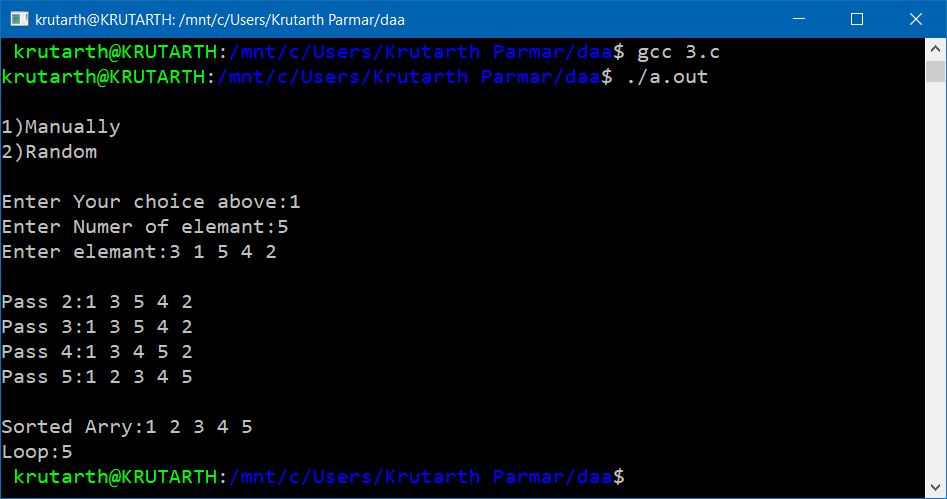
printf("\n\nSorted Arry:"); for(i=0;i<n;i++) printf("%d ",a[i]);

}

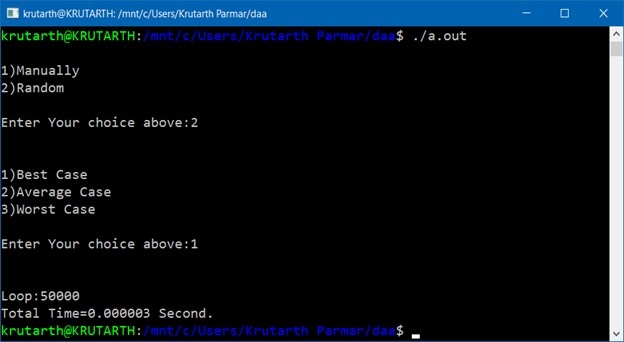
printf("\n\nLoop:%d",lc); printf("\nTotal Time=%lf Second.\n",tt);

}

## Output:



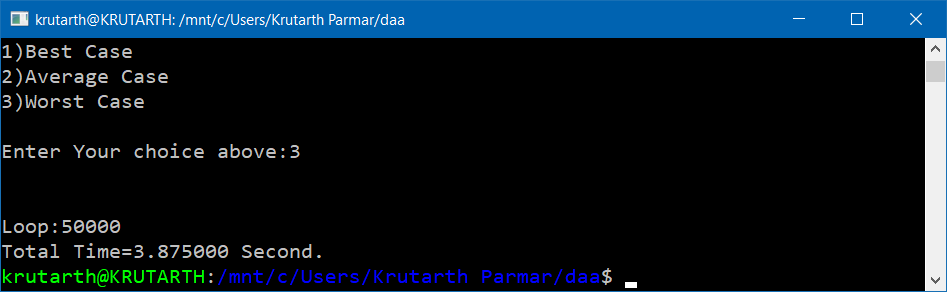
**Best Case:**



**Average Case:**



**Worst Case:**



**Practical 4**

**Aim: Implement quick sort algorithm and perform its best case, average case and worst case analysis.**

**Code:**

#include<stdio.h> #include<time.h>

int a[50000],c,n,m=0,lc=0; int partition(int p,int r)

{

int i,j,x,temp; i=p-1; x=a[r];

for(j=p;lc++,j<r;j++) if(a[j]<=x)

{

i++;

temp=a[i]; a[i]=a[j]; a[j]=temp;

}

temp=a[i+1]; a[i+1]=a[r]; a[r]=temp; if(c==1)

{

}

return i+1;

}

printf("\nPass %d:",++m); for(x=0;x<n;x++) printf("%d ",a[x]);

void quicksort(int p,int r)

{

if(p<r)

{

}

}

int q=partition(p,r); quicksort(p,q-1); quicksort(q+1,r);

void main()

{

clock\_t st,et; int i,j,cc; double tt;

printf("\n1)Manually\n2)Random"); printf("\n\nEnter Your choice above:"); scanf("%d",&c);

if(c==1)

{

printf("Enter Number of Array Element:"); scanf("%d",&n);

printf("Enter Array:"); for(i=0;i<n;i++) scanf("%d",&a[i]); st=clock(); quicksort(0,n-1); et=clock();

}

else if(c==2)

{

n=50000;

printf("\n\n1)Best Case"); printf("\n2)Average Case"); printf("\n3)Worst Case"); printf("\n\nEnter Your choice above:"); scanf("%d",&cc);

switch(cc)

{

case 1:

for(i=0,j=49999;i<n;i++,j--) a[i]=j;

st=clock(); quicksort(0,49999); et=clock();

break;

case 2:

for(i=0;i<n/2;i++) a[i]=i;

for(i=n/2,j=49999;i<n;i++,j--) a[i]=j;

st=clock(); quicksort(0,49999); et=clock();

break;

case 3:

for(i=0;i<n;i++) a[i]=i; st=clock();

quicksort(0,49999); et=clock();

break;

}

}

else

default:

printf("\nWrong Choice\n"); break;

printf("\nWrong Choice\n");

tt=(double) (et-st)/CLOCKS\_PER\_SEC; if(c==1)

{

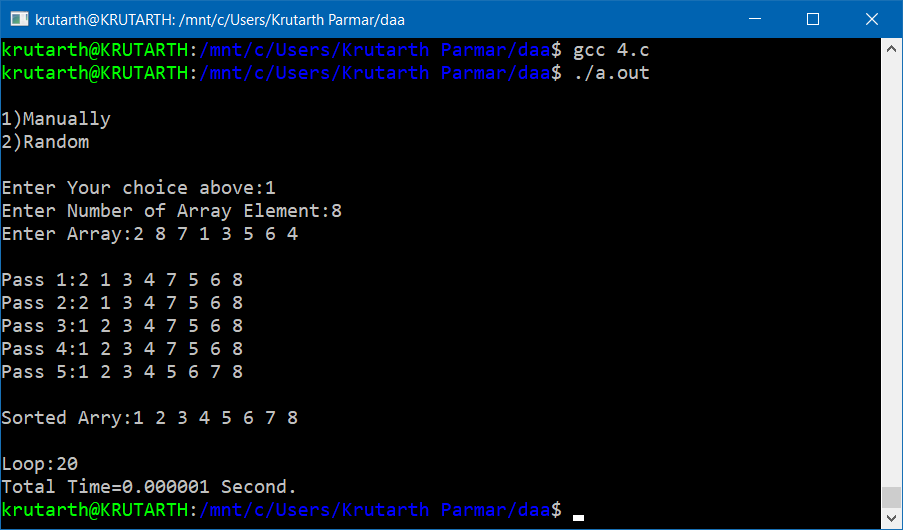
printf("\n\nSorted Arry:"); for(i=0;i<n;i++) printf("%d ",a[i]);

}

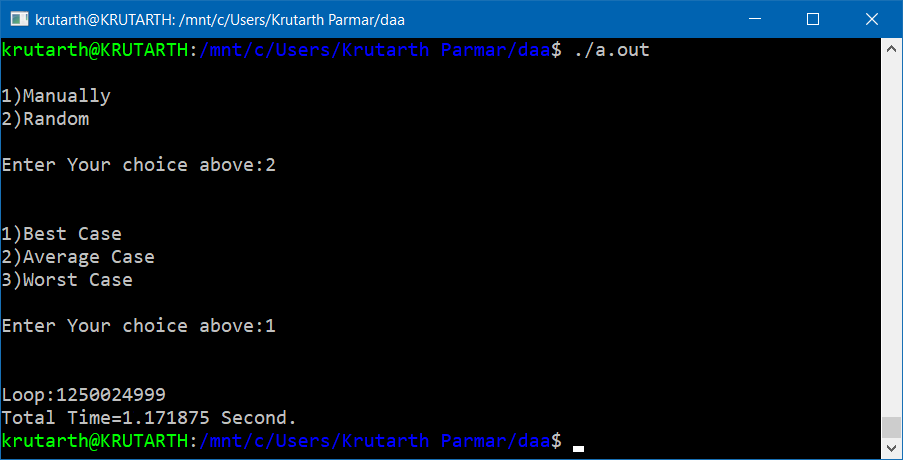
printf("\n\nLoop:%d",lc); printf("\nTotal Time=%lf Second.\n",tt);

}

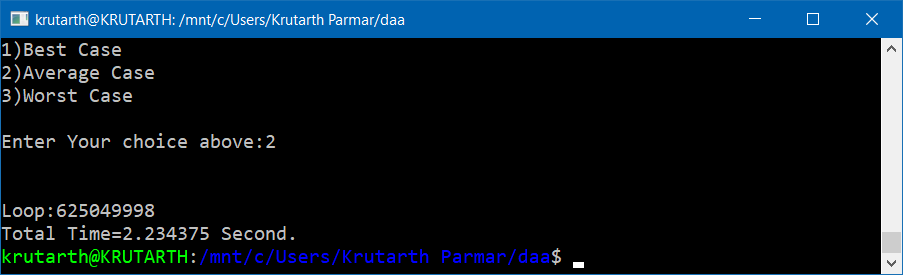
## Output:



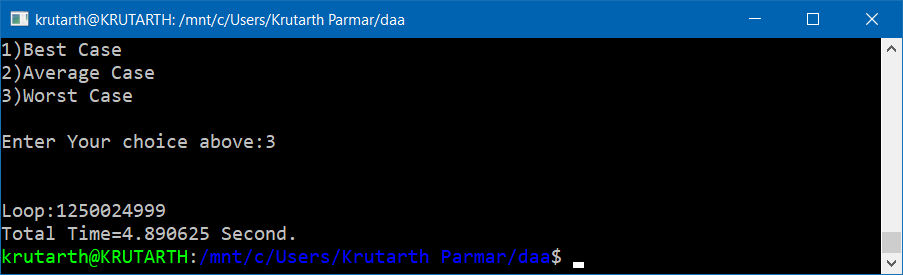
**Best Case:**



**Average Case:**



**Worst Case:**



**Practical 5**

**Aim: Implement knapsack problem using greedy approach. Code:**

#include<stdio.h> int n,i,j;

double tw=0,tv=0,max; struct d

{

int f;

double v,w,k,r;

}d[100];

void knpsacp()

{

double a[100],temp,key; for(i=0;i<n;i++) a[i]=d[i].r; for(i=1;i<n;i++)

{

key=a[i]; j=i-1;

while(a[j]<key && j>=0)

{

a[j+1]=a[j]; j--;

}

a[j+1]=key;

}

for(i=0;i<n;i++) for(j=0;j<n;j++) if(a[i]==d[j].r)

if(d[j].f!=28) if(tw+d[j].w<=max)

{

}

else

{

tw=tw+d[j].w;

tv=tv+d[j].v;

d[j].k=1;

d[j].f=28; j=n;

temp=(max-tw)/d[j].w;

tw=tw+d[j].w\*temp; tv=tv+d[j].v\*temp; d[j].k=temp;

d[j].f=28; i=j=n;

}

}

void main()

{

printf("Enter Maximum weight:"); scanf("%lf",&max);

printf("Enter Number of item:"); scanf("%d",&n); for(i=0;i<n;i++)

{

printf("\n");

printf("Enter weight of %d item:",i+1); scanf("%lf",&d[i].w);

printf("Enter value of %d item:",i+1); scanf("%lf",&d[i].v);

d[i].r=d[i].v/d[i].w; d[i].f=0;

}

printf("\n----------Problem \n");

printf("\n Weight \t Value"); for(i=0;i<n;i++)

printf("\n %.2lf \t %.2lf",d[i].w,d[i].v); printf("\n");

knpsacp();

printf("\n----------Solution \n");

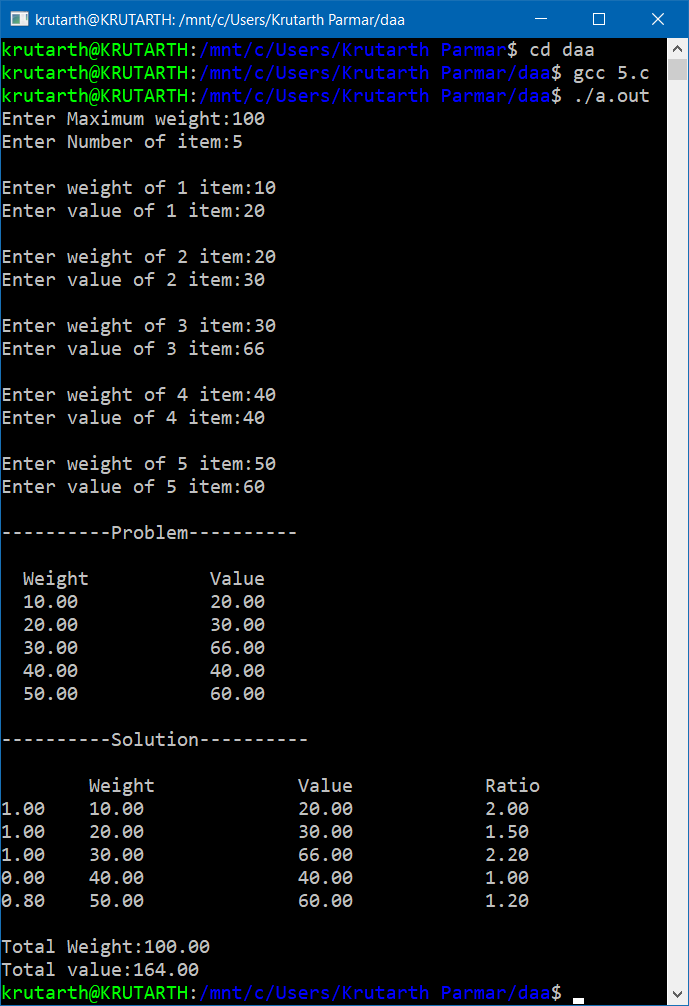
printf("\n\tWeight \t Value \t Ratio"); for(i=0;i<n;i++)

printf("\n%.2lf %.2lf \t %.2lf \t %.2lf",d[i].k,d[i].w,d[i].v,d[i].r); printf("\n\nTotal Weight:%.2lf",tw);

printf("\nTotal value:%.2lf\n",tv);

}

## Output:



**Practical 6**

**Aim: Implement Prim’s algorithm for finding shortest path. Code:**

#include<stdio.h> void main()

{

int i,j,a,b,f,n,min,temp,vnode=1,tcost=0; int c[100][100],root[100]; printf("1)Number\n2)Alphabet"); printf("\n\nEnter Your choice:"); scanf("%d",&temp);

if(temp==1) f=1;

if(temp==2) f=65;

printf("\nEnter Number of Node:"); scanf("%d",&n);

printf("\n"); for(i=0;i<n;i++) for(j=i+1;j<n;j++)

{

if(f==1)

printf("Enter cost of edje[%d][%d]:",i+f,j+f); if(f==65)

printf("Enter cost of edje[%c][%c]:",i+f,j+f); scanf("%d",&c[i][j]);

c[j][i]=c[i][j]; if(c[i][j]==0) c[i][j]=c[j][i]=999;

}

printf("\nEnter Starting Node(Number):"); scanf("%d",&temp);

root[temp-1]=1; while(vnode<n)

{

min=999; for(i=0;i<n;i++) for(j=0;j<n;j++) if(c[i][j]<min)

if(root[i]!=0)

{

min=c[i][j];

a=i; b=j;

}

if(root[a]==0 || root[b]==0)

{

if(f==1)

printf("\nNode %d to %d Cost is:%d",a+f,b+f,min); if(f==65)

printf("\nNode %c to %c Cost is:%d",a+f,b+f,min); tcost+=min;

root[b]=1; vnode++;

}

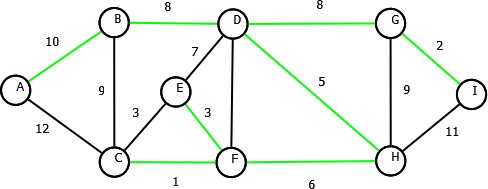
c[a][b]=c[b][a]=999;

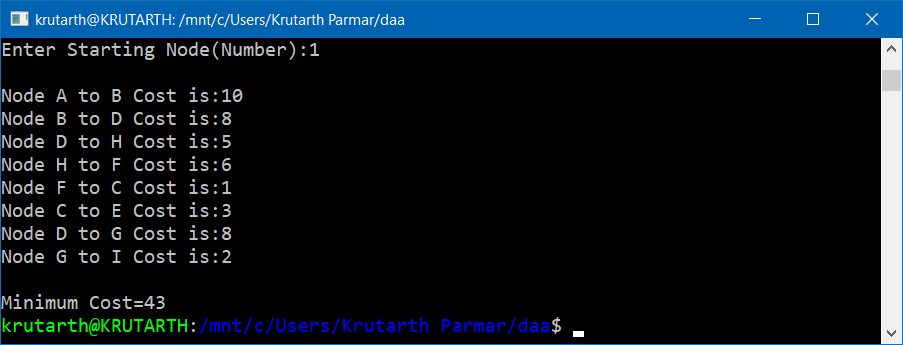
}

printf("\n\nMinimum Cost=%d\n",tcost);

}

## Output:





**Practical 7**

**Aim: Implement making change problem using dynamic programming. Code:**

#include<stdio.h> void main()

{

int i,j,n,N,x,y;

int c[100][100],d[100],m[100];

printf("Enter Number of Coins:"); scanf("%d",&n);

printf("\n"); for(i=1;i<=n;i++)

{

printf("Enter value of %d Coins:",i); scanf("%d",&d[i]);

}

printf("\nEnter Amount:"); scanf("%d",&N);

printf("\n\n "); for(j=0;j<=N;j++) printf("\t %d",j);

printf("\n"); for(i=1;i<=n;i++)

{

printf("\nd%d %d",i,d[i]); for(j=0;j<=N;j++)

if(j==0)

{

printf("\t 0");

c[i][j]=0;

}

else if(i==1 && j<d[i])

{

printf("\tINF");

c[i][j]=999;

}

else if(i==1)

{

else

c[i][j]=1+c[1][j-d[i]]; if(c[i][j]<999) printf("\t %d",c[i][j]);

printf("\tINF");

}

else if(j<d[i])

{

}

else

{

}

}

c[i][j]=c[i-1][j]; if(c[i][j]<999) printf("\t %d",c[i][j]); else

printf("\tINF");

if(c[i-1][j]<(1+c[i][j-d[i]])) c[i][j]=c[i-1][j];

else

c[i][j]=1+c[i][j-d[i]]; if(c[i][j]<999) printf("\t %d",c[i][j]); else

printf("\tINF");

printf("\n\nNumber of coins Required:%d",c[n][N]); x=n;y=N;

while(c[x][y]!=0)

{

while(c[x][y]==c[x-1][y]) x--;

y=y-d[x];

m[x]+=1;

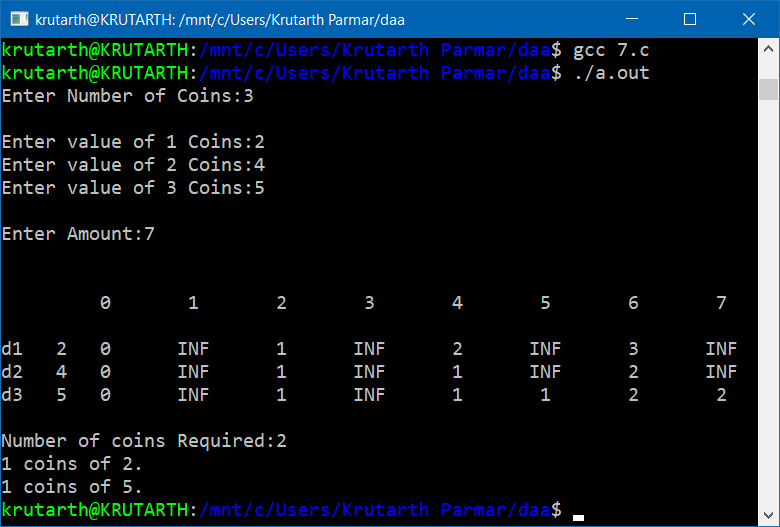
}

for(i=1;i<=n;i++) if(m[i]!=0)

printf("\n%d coins of %d.",m[i],d[i]); printf("\n");

}

## Output:



**Practical 8**

**Aim: Implement Longest common subsequence using dynamic programming.**

**Code:**

#include<stdio.h> #include<string.h> void main()

{

int i,j,k=0;

int c[100][100];

char a[100],b[100],lcs[100]; printf("Enter First String:"); scanf("%s",b);

printf("Enter Second String:"); scanf("%s",a);

printf("\n\n\t \t"); for(i=0;i<strlen(b);i++) printf("\t%c",b[i]); for(i=0;i<=strlen(a);i++)

{

if(i==0) printf("\n\t "); else

printf("\n\t%c",a[i-1]); for(j=0;j<=strlen(b);j++)

{

if(i==0 || j==0)

{

c[i][j]=0; printf("\t%d",c[i][j]);

}

else if(a[i-1]==b[j-1])

{

}

else

{

}

c[i][j]=1+c[i-1][j-1];

printf("\t%d",c[i][j]);

if(c[i][j-1]>c[i-1][j])

c[i][j]=c[i][j-1]; else

c[i][j]=c[i-1][j];

printf("\t%d",c[i][j]);

}

}

printf("\n\nLongest common subsequence:"); i=strlen(a);

j=strlen(b); while(i>0 && j>0) if(a[i-1]==b[j-1])

{

lcs[k]=a[i-1];

i--;j--;k++;

}

else if(c[i-1][j]>c[i][j-1])

{ i--; }

else

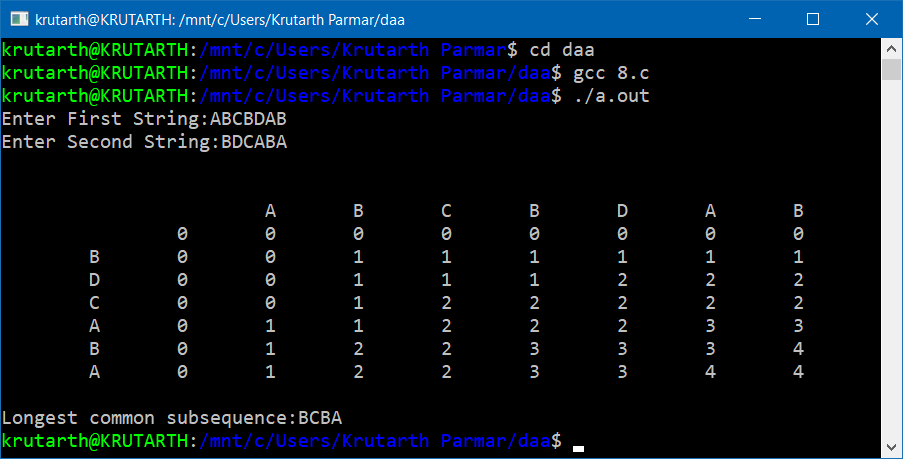
{ j--; }

for(i=strlen(lcs)-1;i>=0;i--)

printf("%c",lcs[i]); printf("\n");

}

## Output:



**Practical 9**

**Aim: Implement topological sorting algorithm and measure its execution time.**

**Code:**

#include<stdio.h> #include<string.h> #include<time.h> void main()

{

clock\_t st,et;

int i,j,k,f,n,m,x=1,lc=0;

char s[100],p[100]; double tt; printf("Enter String:"); scanf("%s",s);

printf("Enter Pattern:"); scanf("%s",p); st=clock();

n=strlen(s); m=strlen(p);

for(i=0;lc++,i<=n-m;i++)

{

f=0;

k=i; for(j=0;lc++,j<m;j++,k++) if(s[k]==p[j])

f++;

else break; if(f==m)

{

}

}

if(x)

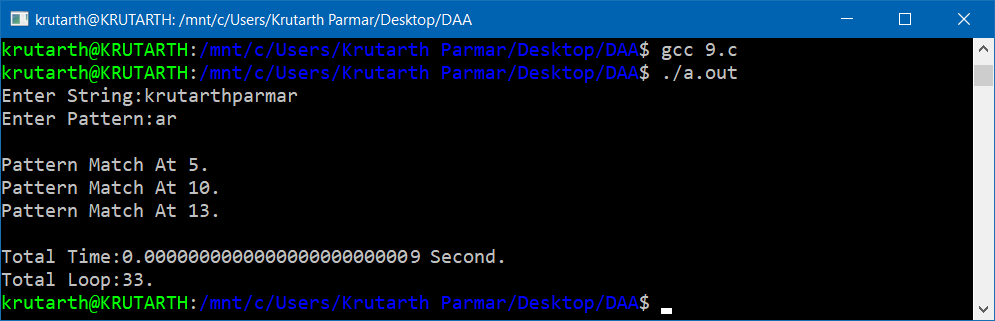
printf("\nPattern Match At %d.",i+1); x=0;

printf("\nNo Match Found."); et=clock();

tt=(double) (et-st)/CLOCKS\_PER\_SEC; printf("\n\nTotal Time:%.25lf Second.",tt); printf("\nTotal Loop:%d.\n",lc);

}

## Output:



**Practical 10**

**Aim: Implement a program that can traverse a path using depth first search algorithm.**

**Code:**

#include<stdio.h>

int a[100][100],r[100],s[100],n,f,k=0;

int check(int x)

{

int i; for(i=0;i<k;i++) if(s[i]==x)

{

}

return x;

}

x=0;

i=k+1;

void dfs(int node)

{

int i; r[node]=1;

for(i=0;i<n;i++) if(a[node][i] && !(r[i]))

{

if(f==1)

printf("\n\t %d --> %d",node+f,i+f); else

printf("\n\t %c --> %c",node+f,i+f); s[k]=check(node+f);

k++;

s[k]=check(i+f); k++;

dfs(i);

}

}

void main()

{

int i,j,c=0,temp; printf("1)Number\n2)Alphabet"); printf("\n\nEnter Your choice:"); scanf("%d",&temp);

if(temp==1) f=1;

if(temp==2) f=65;

printf("\nEnter Number of Node:"); scanf("%d",&n);

printf("\n"); for(i=0;i<n;i++) for(j=i+1;j<n;j++)

{

if(f==1)

printf("Edge between Node[%d][%d]:",i+f,j+f); else

printf("Edge between Node[%c][%c]:",i+f,j+f); scanf("%d",&a[i][j]);

}

printf("\n\*\*\*\*\* Selected Edge \*\*\*\*\*"); dfs(0);

for(i=0;i<n;i++) if(r[i])

c++;

printf("\n\n\t"); for(i=0;i<k;i++) if(s[i]!=0) if(f==1) printf("%d ",s[i]); else

printf("%c ",s[i]); if(c!=n)

printf("\nGrap is Not Connected."); printf("\n");

}

## Output:

