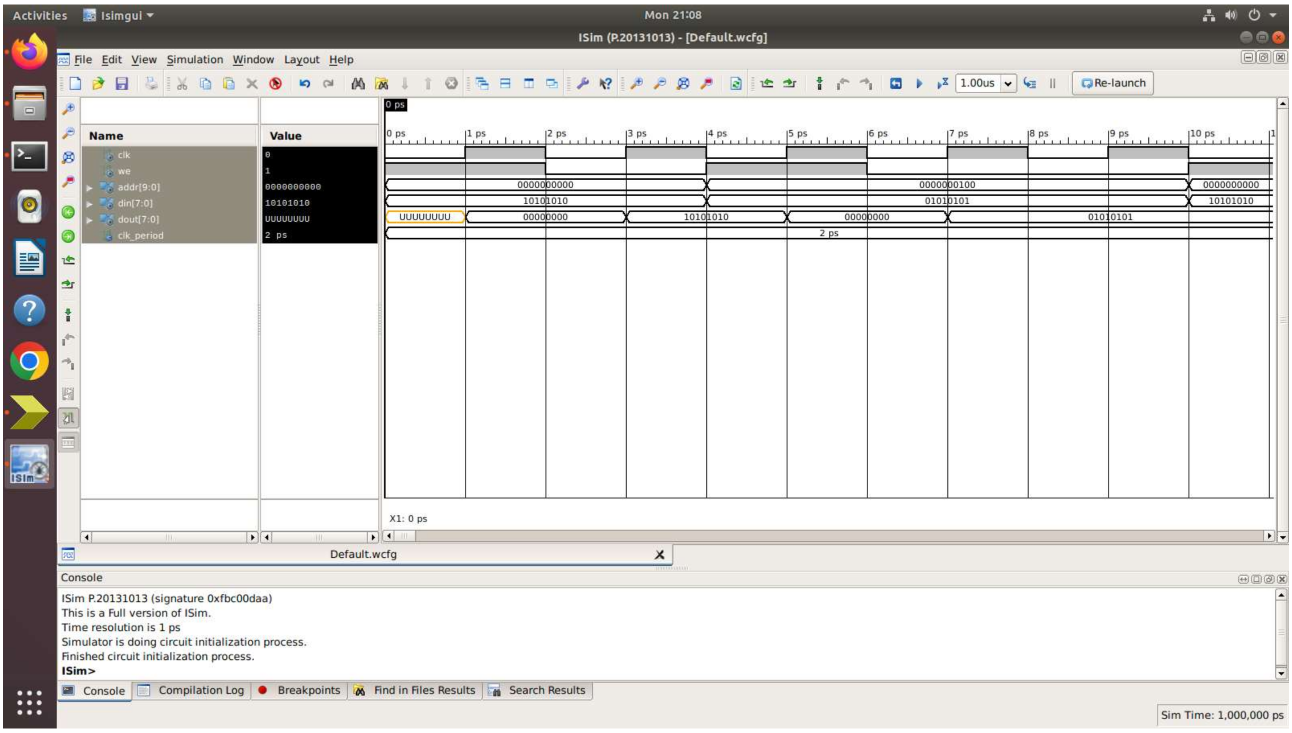
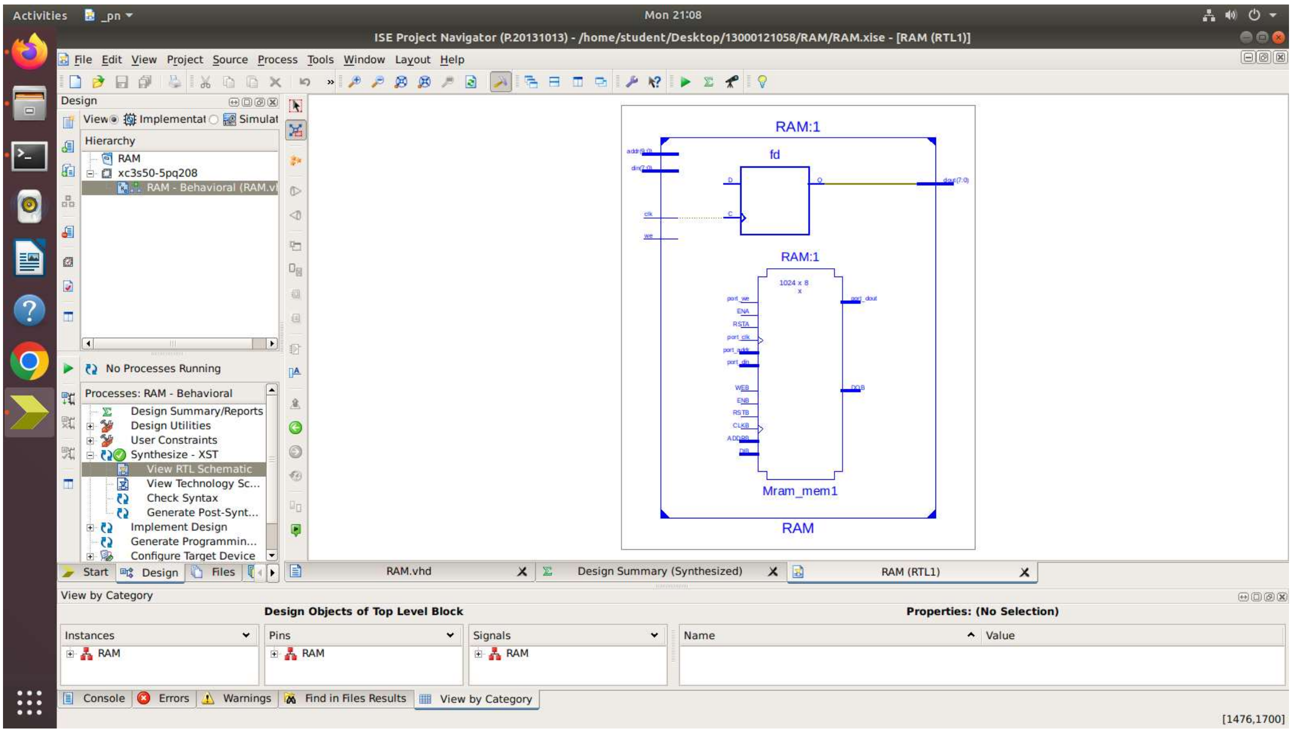
TEST OUTPUT



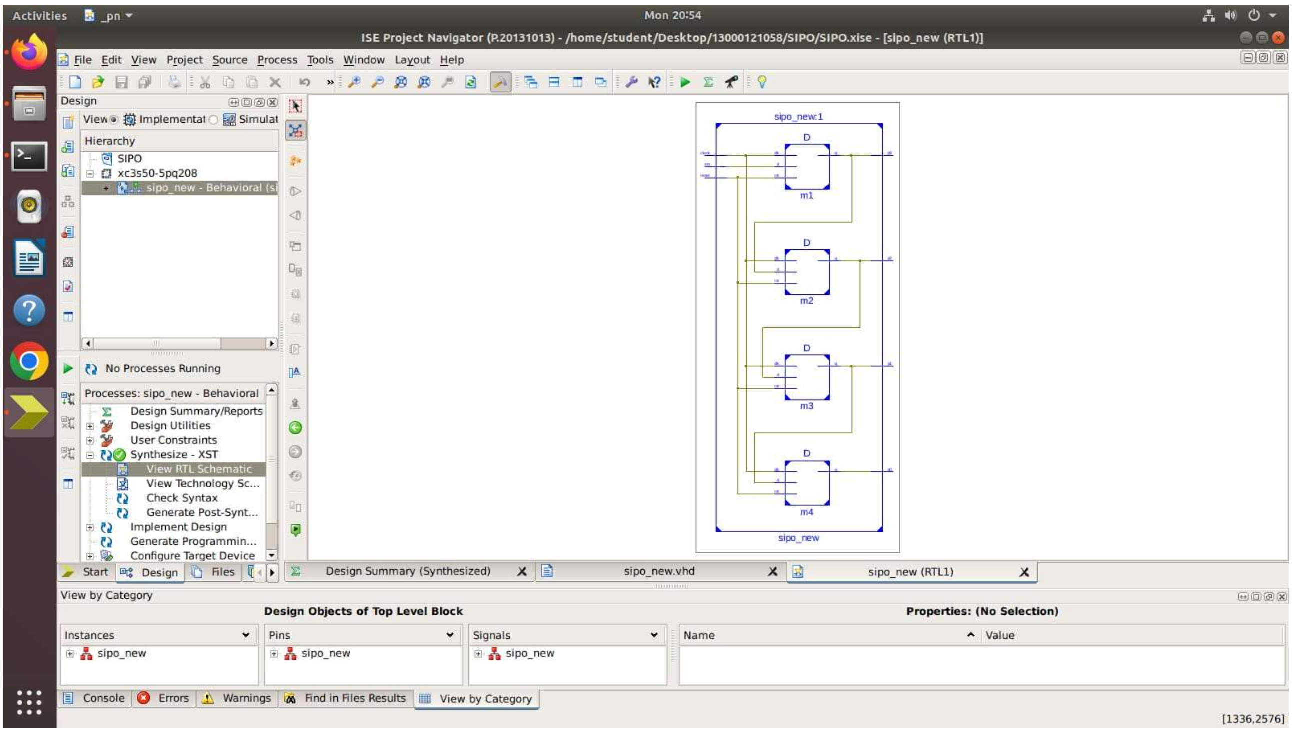
Name	Value
clk	0
we	1
addr[9:0]	0000000000
din[7:0]	10101010
dout[7:0]	UUUUUUUU
clk_period	2 ps

```
ISim P.20131013 (signature 0xfbc00daa)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

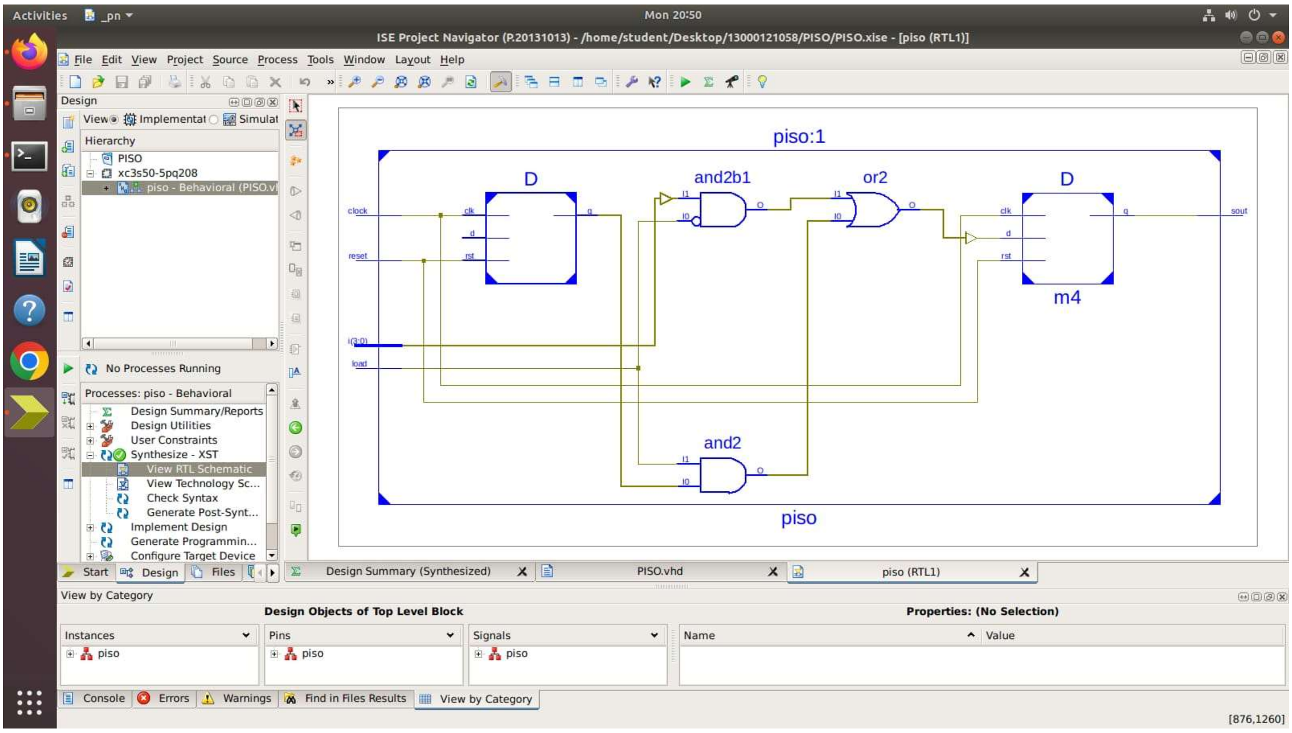
SCHEMATIC OUTPUT



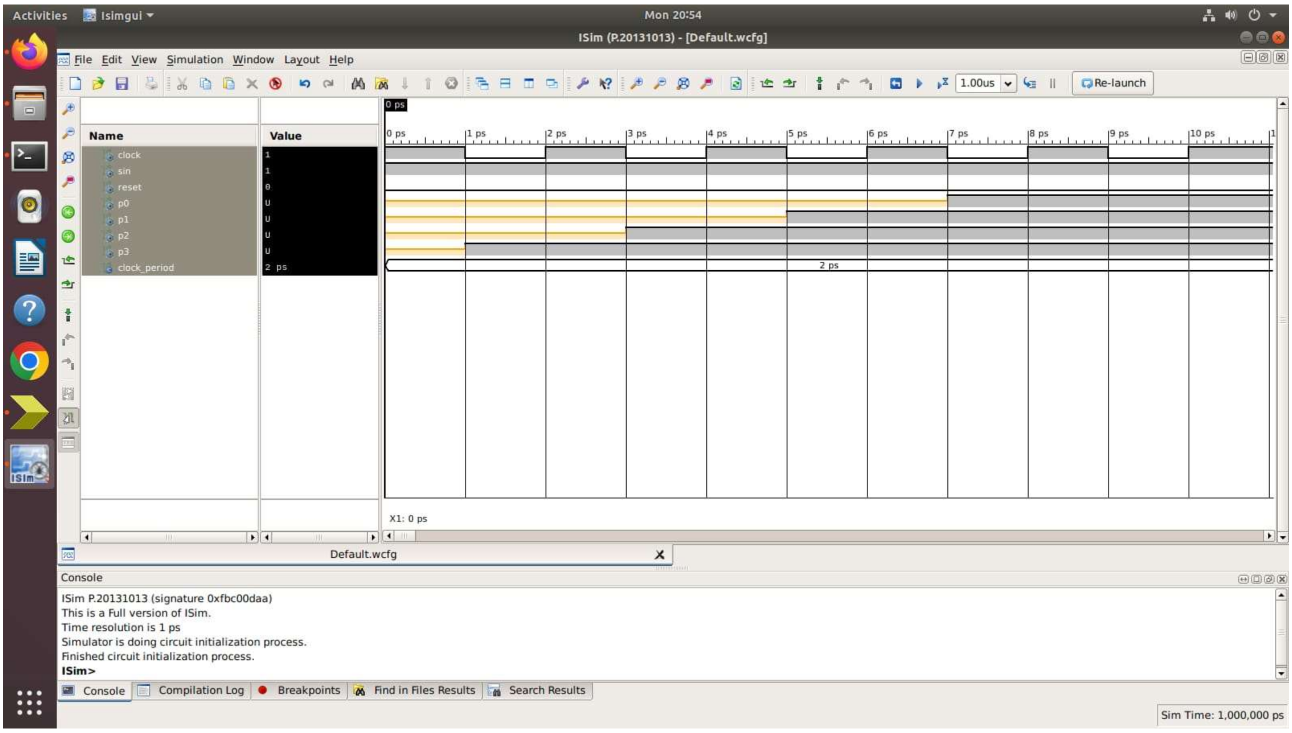
SCHEMATIC OUTPUT



SCHEMATIC OUTPUT



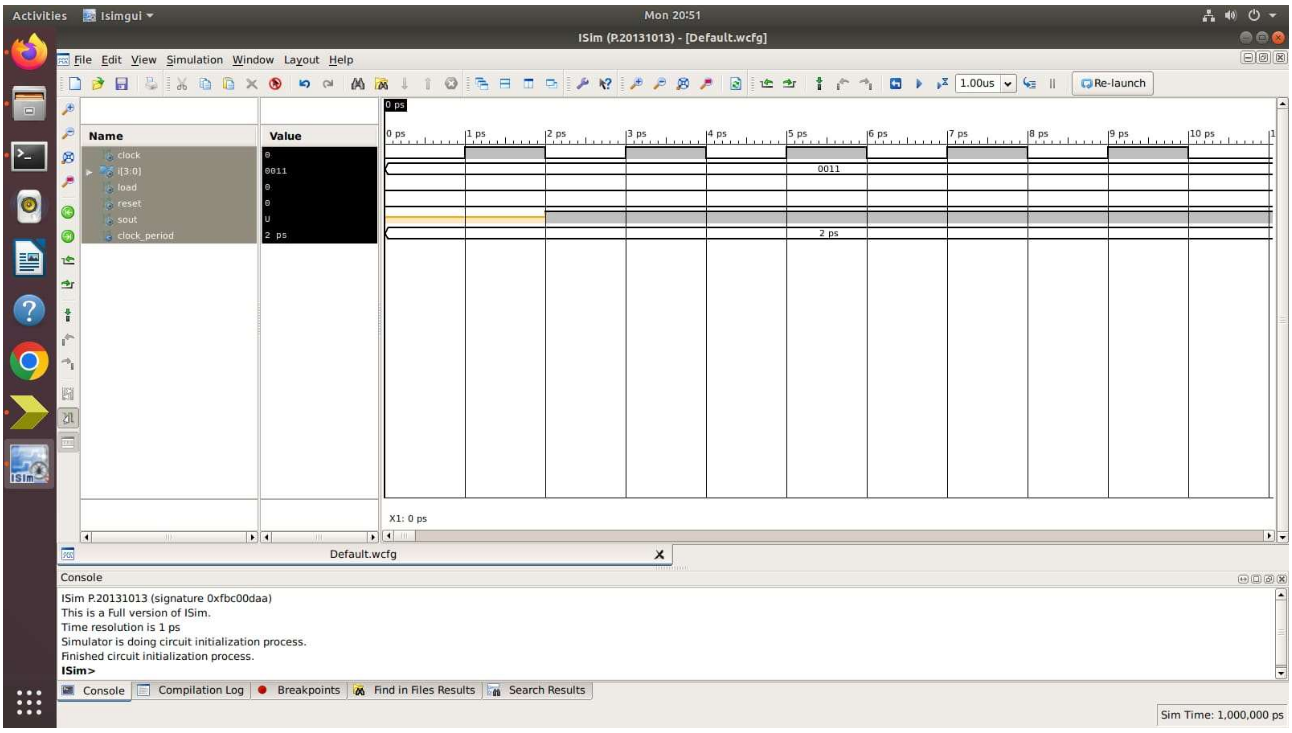
TEST OUTPUT



Name	Value
clock	1
sin	1
reset	0
p0	U
p1	U
p2	U
p3	U
clock_period	2 ps

```
ISim P.20131013 (signature 0xfbc00daa)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

TEST OUTPUT

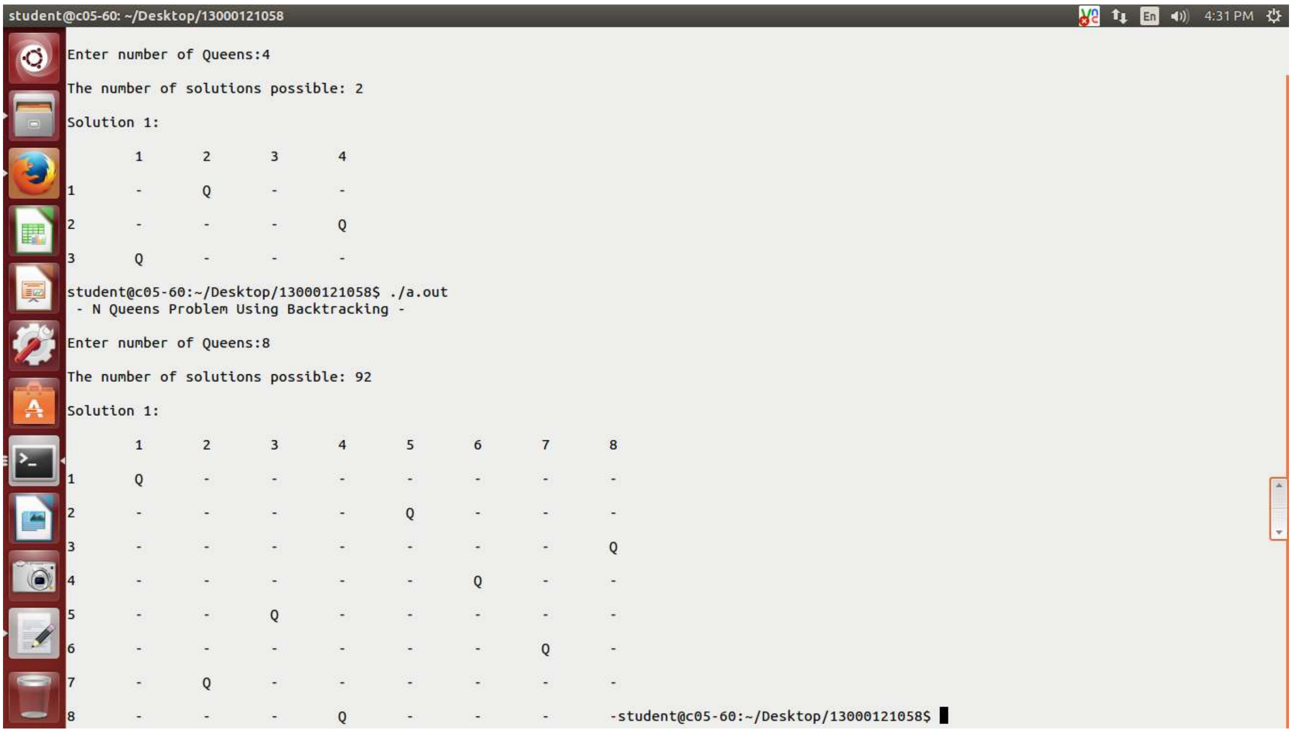


Name	Value
clock	0
i[3:0]	0011
load	0
reset	0
sout	U
clock_period	2 ps

Console

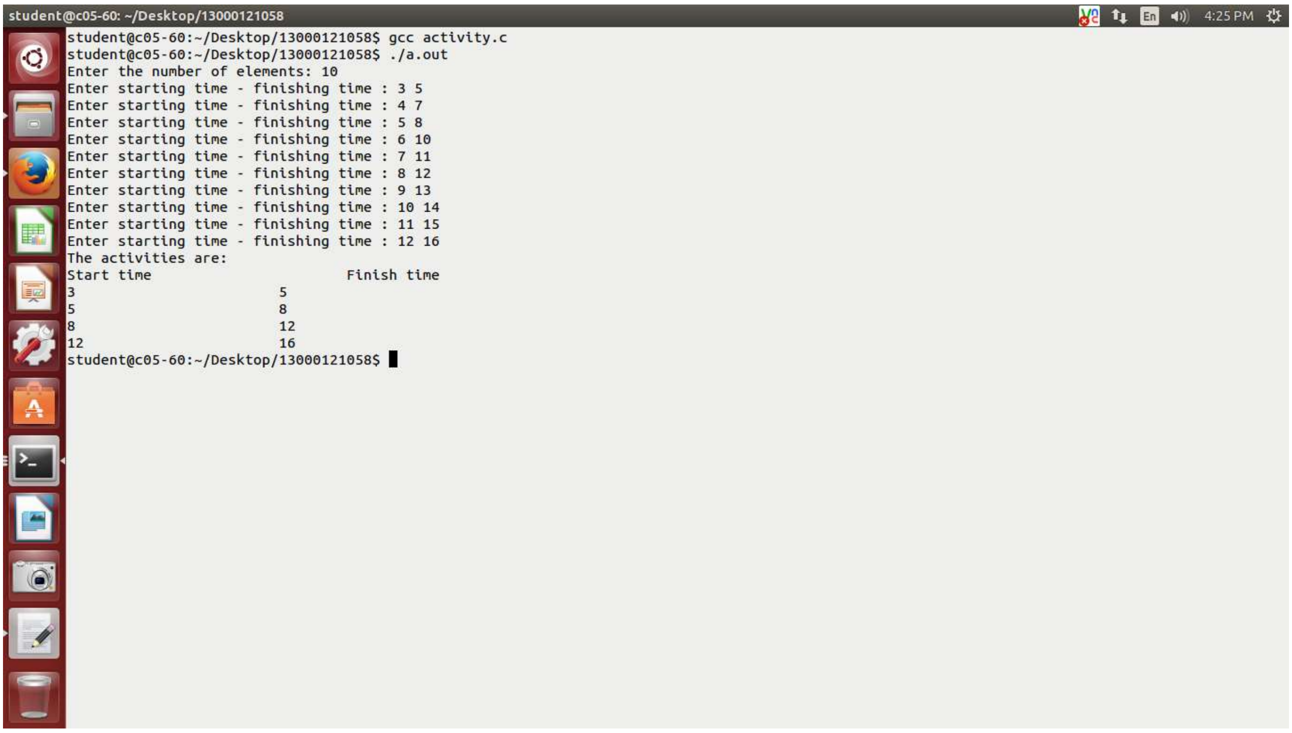
```
ISim P.20131013 (signature 0xfbc00daa)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

OUTPUT



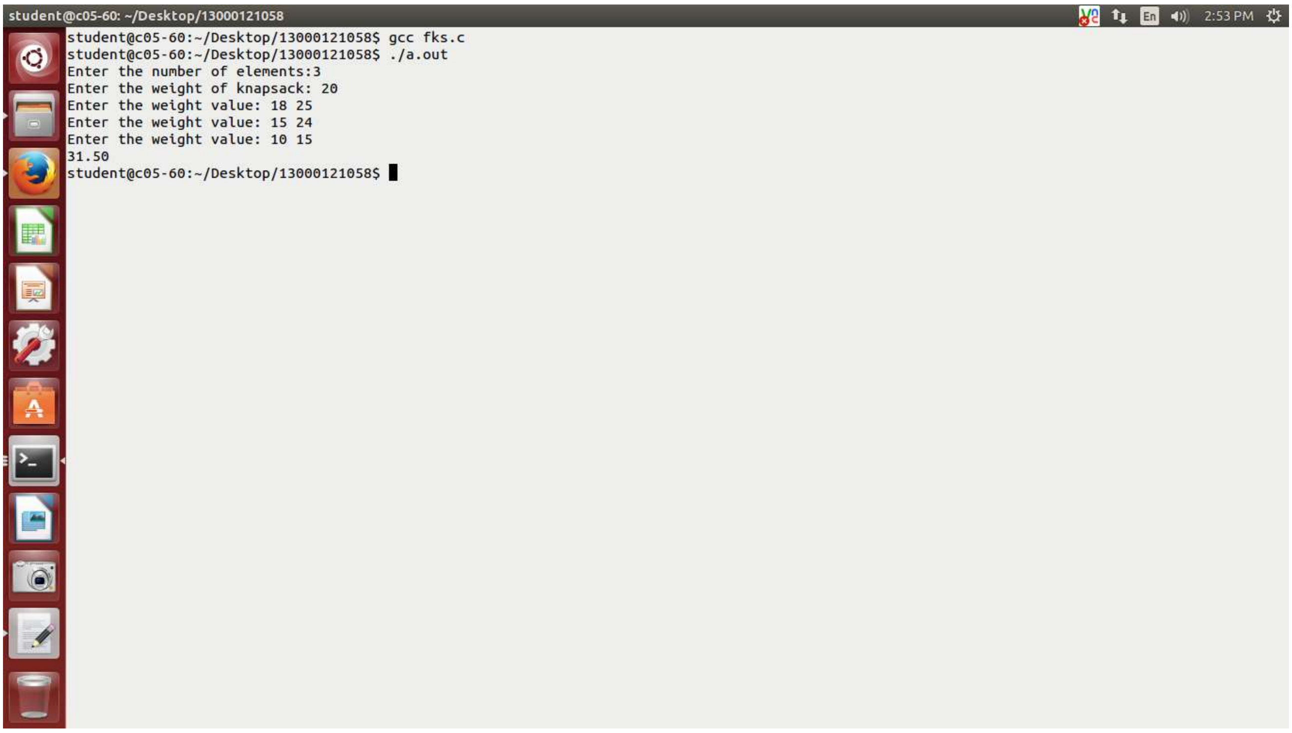
```
Enter number of Queens:4

The number of solutions possible: 2

Solution 1:

        1       2       3       4

1       -       Q       -       -

2       -       -       -       Q

3       Q       -       -       -

student@c05-60:~/Desktop/13000121058$ ./a.out
 - N Queens Problem Using Backtracking -

Enter number of Queens:8

The number of solutions possible: 92

Solution 1:

        1       2       3       4       5       6       7       8

1       Q       -       -       -       -       -       -       -

2       -       -       -       -       Q       -       -       -

3       -       -       -       -       -       -       -       Q

4       -       -       -       -       -       Q       -       -

5       -       -       Q       -       -       -       -       -

6       -       -       -       -       -       -       Q       -

7       -       Q       -       -       -       -       -       -

8       -       -       -       Q       -       -       -       -student@c05-60:~/Desktop/13000121058$ 
```

OUTPUT



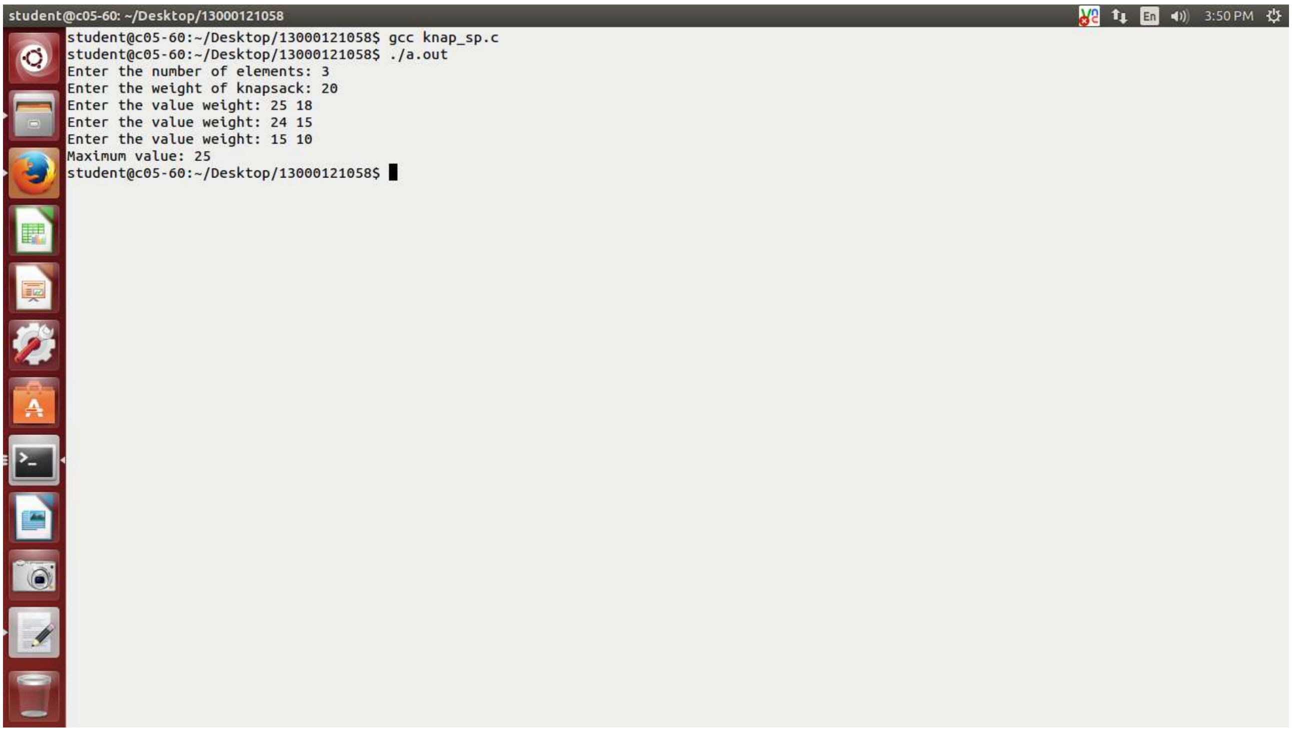
```
student@c05-60:~/Desktop/13000121058$ gcc activity.c
student@c05-60:~/Desktop/13000121058$ ./a.out
Enter the number of elements: 10
Enter starting time - finishing time : 3 5
Enter starting time - finishing time : 4 7
Enter starting time - finishing time : 5 8
Enter starting time - finishing time : 6 10
Enter starting time - finishing time : 7 11
Enter starting time - finishing time : 8 12
Enter starting time - finishing time : 9 13
Enter starting time - finishing time : 10 14
Enter starting time - finishing time : 11 15
Enter starting time - finishing time : 12 16
The activities are:
Start time                       Finish time
3                        5
5                        8
8                        12
12                       16
student@c05-60:~/Desktop/13000121058$ 
```

OUTPUT



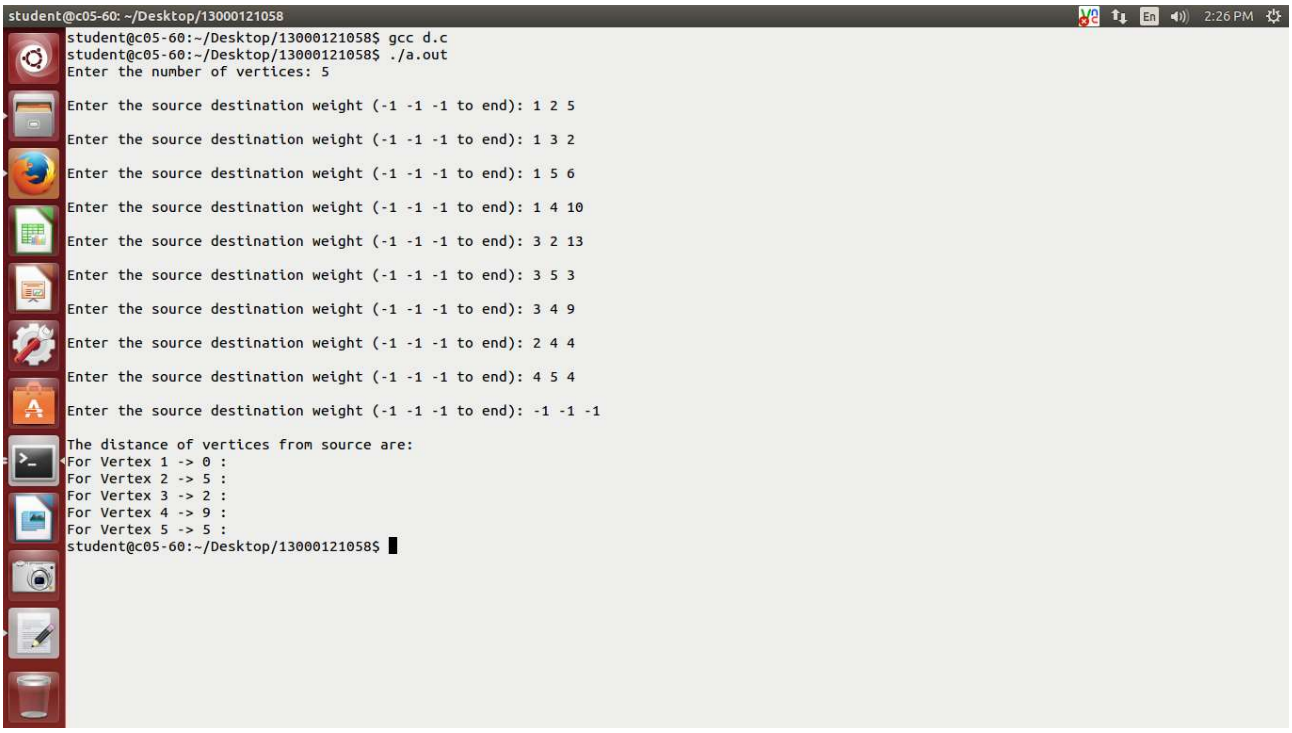
```
student@c05-60:~/Desktop/13000121058$ gcc fks.c
student@c05-60:~/Desktop/13000121058$ ./a.out
Enter the number of elements:3
Enter the weight of knapsack: 20
Enter the weight value: 18 25
Enter the weight value: 15 24
Enter the weight value: 10 15
31.50
student@c05-60:~/Desktop/13000121058$
```

OUTPUT



```
student@c05-60:~/Desktop/13000121058$ gcc knap_sp.c
student@c05-60:~/Desktop/13000121058$ ./a.out
Enter the number of elements: 3
Enter the weight of knapsack: 20
Enter the value weight: 25 18
Enter the value weight: 24 15
Enter the value weight: 15 10
Maximum value: 25
student@c05-60:~/Desktop/13000121058$ 
```

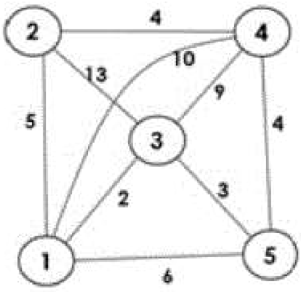
OUTPUT



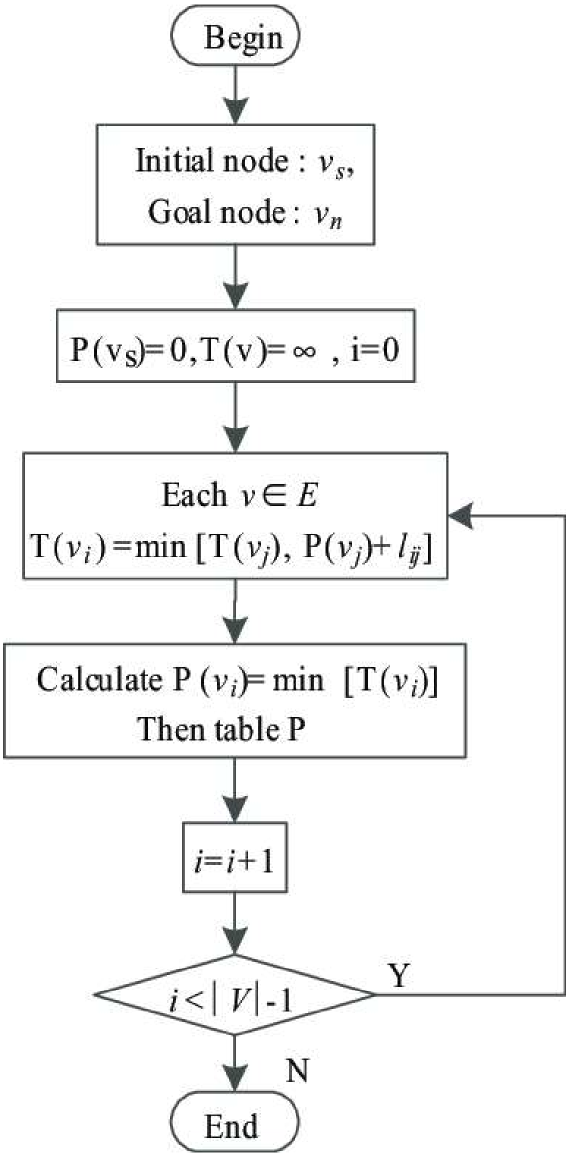
```
student@c05-60:~/Desktop/13000121058$ gcc d.c
student@c05-60:~/Desktop/13000121058$ ./a.out
Enter the number of vertices: 5

Enter the source destination weight (-1 -1 -1 to end): 1 2 5

Enter the source destination weight (-1 -1 -1 to end): 1 3 2

Enter the source destination weight (-1 -1 -1 to end): 1 5 6

Enter the source destination weight (-1 -1 -1 to end): 1 4 10

Enter the source destination weight (-1 -1 -1 to end): 3 2 13

Enter the source destination weight (-1 -1 -1 to end): 3 5 3

Enter the source destination weight (-1 -1 -1 to end): 3 4 9

Enter the source destination weight (-1 -1 -1 to end): 2 4 4

Enter the source destination weight (-1 -1 -1 to end): 4 5 4

Enter the source destination weight (-1 -1 -1 to end): -1 -1 -1

The distance of vertices from source are:
For Vertex 1 -> 0 :
For Vertex 2 -> 5 :
For Vertex 3 -> 2 :
For Vertex 4 -> 9 :
For Vertex 5 -> 5 :
student@c05-60:~/Desktop/13000121058$ 
```

ASSIGNMENT 4.1

Implement Single Source shortest Path for a graph (Dijkstra Algorithm)problem to find out the shortest path from the source vertex ‘1’, using the Dijkstra’s algorithm, using the dynamic programming technique.



FLOWCHART

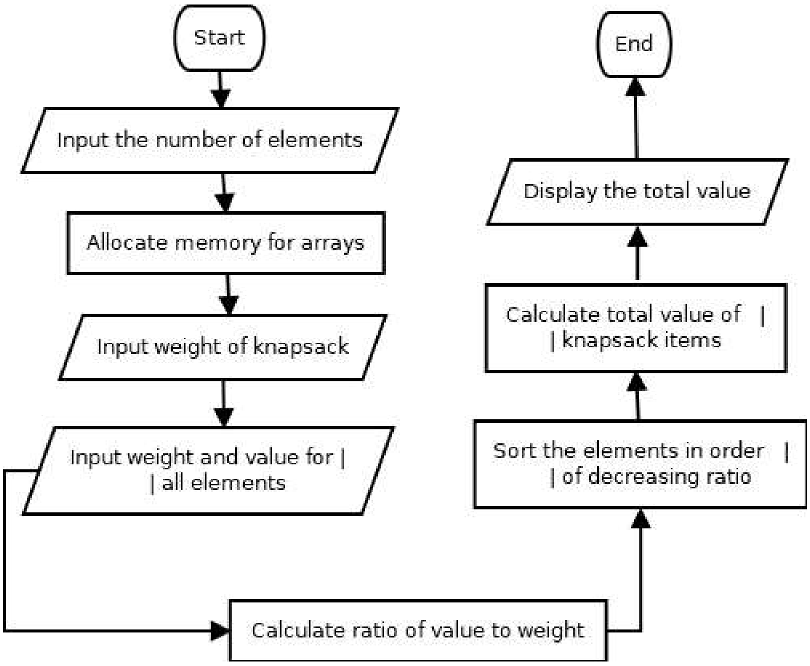

Begin
Initial node : v_s,
Goal node : v_n
$P(v_S)=0, T(v)=\infty$, $i=0$
Each $v \in E$
$T(v_i)=\min[T(v_j), P(v_j)+l_{ij}]$
Calculate $P(v_i)=\min [T(v_i)]$
Then table P
$i=i+1$
$i<|V|-1$
Y
N
End

ASSIGNMENT 8.1

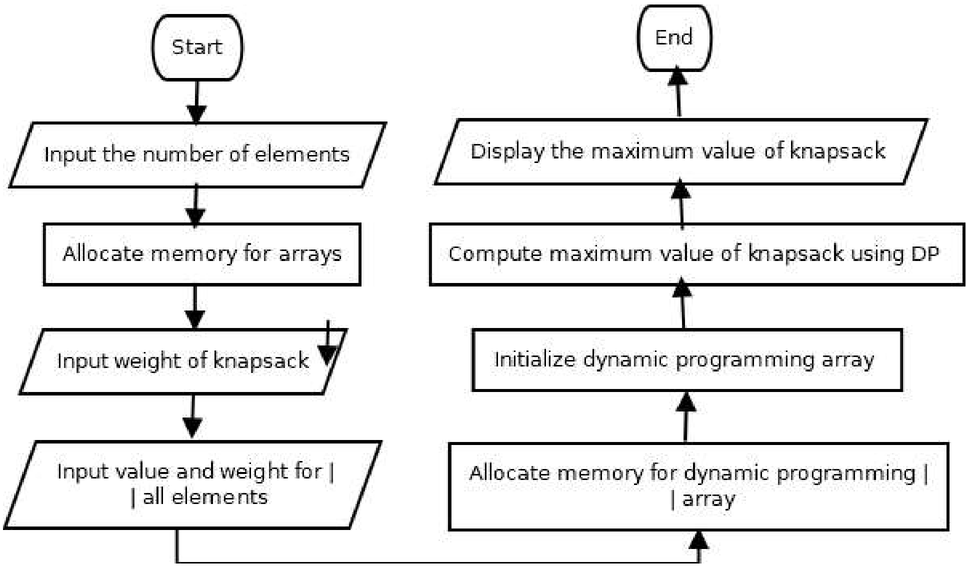
Consider the following knapsack problem where n = 3, W = 20 Kgs, (v1, v2, v3) = (25, 24, 15), and (w1, w2, w3) = (18, 15, 10). WAP to find the optimal solution by fractional knapsack (greedy method) as well as 0 / 1 knapsack (Dynamic programming method).

FLOWCHART

Fractional Knapsack



Start
Input the number of elements
Allocate memory for arrays
Input weight of knapsack
Input weight and value for | | all elements
Calculate ratio of value to weight
Sort the elements in order | | of decreasing ratio
Calculate total value of | | knapsack items
Display the total value
End



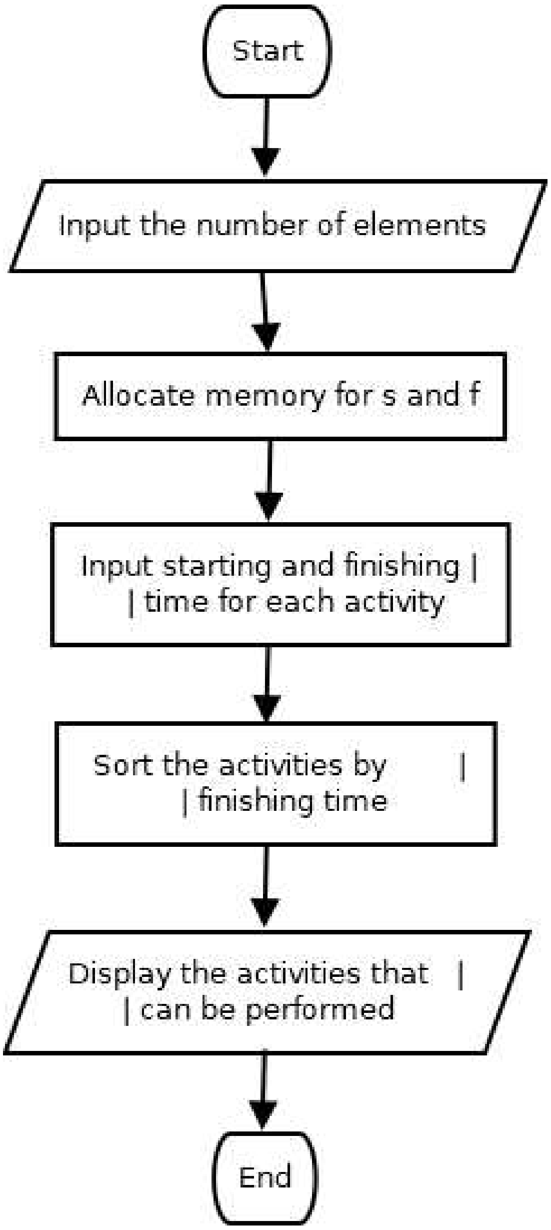
Start
Input the number of elements
Allocate memory for arrays
Input weight of knapsack
Input value and weight for | | all elements
Allocate memory for dynamic programming | | array
Initialize dynamic programming array
Compute maximum value of knapsack using DP
Display the maximum value of knapsack
End

0/1 Knapsack

ASSIGNMENT 8.2

Given a set of '10' jobs with their si and fi, find the optimal sequence of mutually compatible jobs using the greedy method: A = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, si = {3, 4, 5, 6, 7, 8, 9, 10, 11, 12} and fi = {5, 7, 8, 10, 11, 12, 13, 14, 15, 16}.

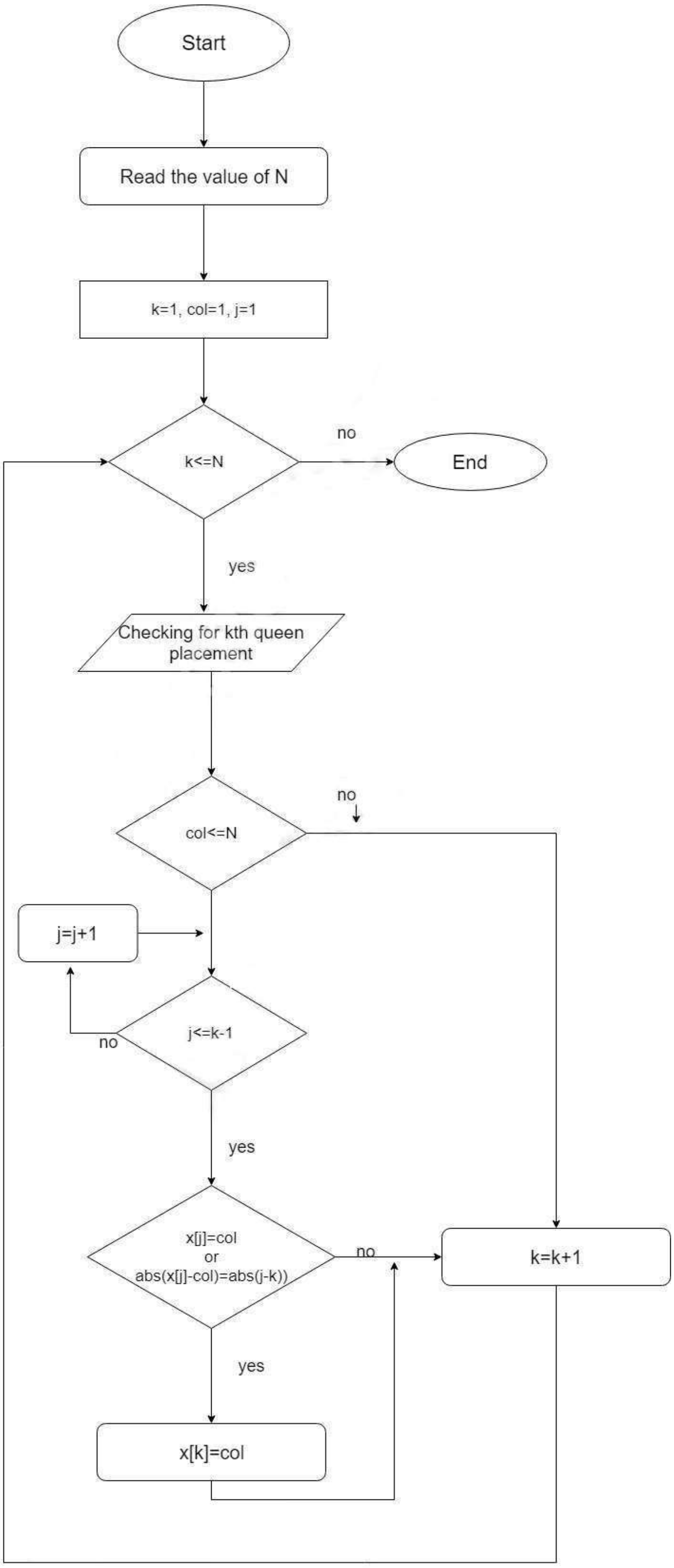
FLOWCHART


Start
Input the number of elements
Allocate memory for s and f
Input starting and finishing |
| time for each activity
Sort the activities by |
| finishing time
Display the activities that |
| can be performed
End

ASSIGNMENT 10.1

WAP to implement the N-Queen's problem using the method of backtracking.

FLOWCHART



N_QUEENS.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include<ctype.h>
int board[20],count;
int counter(int row,int n,int *c);
int place(int row,int column);
void print(int n);
void queen(int row,int n);
int counter(int row, int n,int *c)
{
    int column;
    for (column = 1; column <= n; ++column)
    {
        if (place(row, column))
        {
            board[row] = column;
            if (row == n)
               (*c)+=1;//print(n);
            else
                counter(row + 1, n,c);
        }
    }
}
int place(int row, int column)
{
    int i;
    for (i = 1; i <= row - 1; ++i)
    {
        if (board[i] == column)
            return 0;
        else if (fabs(board[i] - column) == fabs(i - row))
            return 0;
    }

    return 1;

}
void queen(int row, int n)
{
    int column;
    for (column = 1; column <= n; ++column)
    {
        if (place(row, column))
        {
            board[row] = column;
            if (row == n)
               print(n);
            else
                queen(row + 1, n);
        }
    }
}
void print(int n)
{
    int i, j;
    printf("\n\nSolution %d:\n\n", ++count);

    for (i = 1; i <= n; ++i)
        printf("\t%d", i);

    for (i = 1; i <= n; ++i)
    {
        printf("\n\n%d", i);
        for (j = 1; j <= n; ++j)
        {
            if (board[i] == j)
                printf("\tQ");
            else
                printf("\t-");
        }
    }
    if(count==1)
    {
        exit(0);
    }
}
int main()
{
    int n, i, j,c=0;
    void queen(int row, int n);
    printf(" - N Queens Problem Using Backtracking -");
    printf("\n\nEnter number of Queens:");
    fflush(stdin);
    scanf("%d", &n);
    counter(1,n,&c);
    printf("\nThe number of solutions possible: %d",c);
    queen(1, n);
    return 0;
}
```

knap_sp.c

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n,*v,*wt,*p,w,i,j,max=0;
    int nottake,take;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    fflush(stdin);
    v=(int *)malloc(n*sizeof(int));
    wt=(int *)malloc(n*sizeof(int));
    printf("Enter the weight of knapsack: ");
    scanf("%d",&w);
    fflush(stdin);
    for(i=0;i<n;i++)
    {
        printf("Enter the value weight: ");
        scanf("%d %d",&v[i],&wt[i]);
        fflush(stdin);
    }
    p=(int *)malloc((w+1)*sizeof(int));
    for(i=0;i<=w;i++)
        p[i]=-1;
    for(i=wt[0];i<=w;i++)p[i]=v[0];
    for(i=1;i<n;i++)
    {
        for(j=w;j>=0;j--)
        {
            nottake=0+p[j];
            take=-9999;
            if(wt[i]<=j)
                take=v[i]+p[j-wt[i]];
            p[j]=(take>nottake)?take:nottake;
        }
    }
    //max=f(v,wt,w,n-1,dp);
    printf("Maximum value: %d\n",p[w]);
}
```

fks.c

```c
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n,i,j,c;
    float *r,*wt,*v,w,o,d,temp,rem,current=0,totalval=0;
    printf("Enter the number of elements:");
    scanf("%d",&n);
    fflush(stdin);
    v=(float *)malloc(n*sizeof(float));
    wt=(float *)malloc(n*sizeof(float));
    r=(float *)malloc(n*sizeof(float));
    printf("Enter the weight of knapsack: ");
    scanf("%f",&w);
    fflush(stdin);
    for(i=0;i<n;i++)
    {
        printf("Enter the weight value: ");
        scanf("%f %f",&o,&d);
        fflush(stdin);
        wt[i]=o;
        v[i]=d;
        r[i]=v[i]/wt[i];
    }
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(r[i]>r[i+1])
            {
                temp=r[i];
                r[i]=r[i+1];
                r[i+1]=temp;

                temp=wt[i];
                wt[i]=wt[i+1];
                wt[i+1]=temp;

                temp=v[i];
                v[i]=v[i+1];
                v[i+1]=temp;
            }
        }
    }
    for(i=0;i<n;i++)
    {
        if(current+wt[i]<=w)
        {
            current+=wt[i];
            totalval+=v[i];
        }
        else
        {
            rem=w-current;
            totalval+=rem*r[i];
            break;
        }
    }
    printf("%.2f \n",totalval);
}
```

d.c

```
#include<stdio.h>
#include<stdlib.h>
int mindis(int *dist,int *spt,int n)
{
    int min=9999,minindex=-1,i;
    for(i=0;i<n;i++)
    {
        if(spt[i]==-1 && dist[i]<min)
        {
            min=dist[i];
            minindex=i;
        }
    }
    return minindex;
}
void main()
{
    int n,**g,*dist,count,v,u,*spt,i,o,d,wt,maxedges;
    printf("Enter the number of vertices: ");
    scanf("%d",&n);
    fflush(stdin);
    g=(int **)malloc(n*sizeof(int *));
    for(i=0;i<n;i++)
    {
        g[i]=(int *)malloc(n*sizeof(int));
    }
    maxedges=n*(n-1)/2;
    for(i=0;i<maxedges;i++)
    {
        printf("\nEnter the source destination weight (-1 -1 -1 to end): ");
        scanf("%d %d %d",&o,&d,&wt);
        fflush(stdin);
        if(o==-1 && d==-1 && wt==-1)break;
        g[o-1][d-1]=wt;
        g[d-1][o-1]=wt;
    }

    dist=(int *)malloc(n*sizeof(int));
    spt=(int *)malloc(n*sizeof(int));
    for(i=0;i<n;i++)
    {
        dist[i]=9999;
        spt[i]=-1;
    }
    dist[0]=0;
    for(count=0;count<n-1;count++)
    {
        u=mindis(dist,spt,n);
        spt[u]=1;
        for(v=0;v<n;v++)
        {
            if(spt[v]==-1 && g[u][v] && dist[u]!=9999 && dist[u]+g[u][v]<dist[v])
                dist[v]=dist[u]+g[u][v];
        }
    }
    printf("\nThe distance of vertices from source are:\n");
    for(i=0;i<n;i++)
    {
        printf("For Vertex %d -> %d : \n",(i+1),dist[i]);
    }
}
```

activity.c

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n,*s,*f,i,j,temp=0,c=1;;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    fflush(stdin);
    s=(int *)malloc(n*sizeof(int));
    f=(int *)malloc(n*sizeof(int));
    for(i=0;i<n;i++)
    {
        printf("Enter starting time - finishing time : ");
        scanf("%d %d",&s[i],&f[i]);
        fflush(stdin);
    }
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(f[i]>f[i+1])
            {
                temp=f[i];
                f[i]=f[i+1];
                f[i+1]=temp;

                temp=s[i];
                s[i]=s[i+1];
                s[i+1]=temp;
             }
         }
     }
     printf("The activities are: \n");
     printf("Start time \t\t\t Finish time\n");
     printf("%d \t\t\t %d\n",s[0],f[0]);
     i=0;
     for(j=1;j<n;j++)
     {
        if(s[j]>=f[i])
        {
            printf("%d \t\t\t %d\n",s[j],f[j]);
            i=j;
        }
     }
}
```