Source Code:

```
#include <stdio.h>
// C recursive function to solve tower of hanoi puzzle
void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod)
{
    if (n == 1)
    {
     printf("\n Move disk 1 from rod %c to rod %c", from_rod, to_rod);
     return;
     }
     towerOfHanoi(n-1, from_rod, aux_rod, to_rod);
     printf("\n Move disk %d from rod %c to rod %c", n, from_rod, to_rod);
     towerOfHanoi(n-1, aux_rod, to_rod, from_rod);
}

     int main()
{
     int n = 3; // Number of disks
     towerOfHanoi(n, 'P', 'R', 'Q'); // P, Q and R are names of rods
     return 0;
}
```

Output:

```
Move disk 1 from rod P to rod R

Move disk 2 from rod P to rod Q

Move disk 1 from rod R to rod Q

Move disk 3 from rod P to rod R

Move disk 1 from rod Q to rod P

Move disk 2 from rod Q to rod R

Move disk 1 from rod P to rod R
```

Source Code:

```
#include<stdio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
void bfs(int v)
{
   for(i = 1; i <= n; i++)
      if(a[v][i] && !visited[i])
         q[++r] = i;
   if(f <= r)
   {
      visited[q[f]] = 1;
      bfs(q[f++]);
   }
}
void main()
{
   int v;
   printf("\n Enter the number of vertices:");
   scanf("%d", &n);
   for(i=1; i <= n; i++)
   {
      q[i] = 0;
      visited[i] = 0;
   }
   printf("\n Enter graph data in matrix form:\n");
   for(i=1; i<=n; i++)
   {
      for(j=1; j<=n; j++)
      {
```

```
            scanf("%d", &a[i][j]);
        }
    }
    printf("\n Enter the starting vertex:");
    scanf("%d", &v);
    bfs(v);
    printf("\n The node which are reachable are:\n");
    for(i=1; i <= n; i++)
    {
        if(visited[i]) printf("%d\t", i);
        else
        {
            printf("\n Bfs is not possible. Not all nodes are reachable");
            break;
        }
    }
}
```

Input and Output:

```
Enter the number of vertices: 3
Enter graph data in matrix form:
 0 1 1
 0 1 0
 1 1 0
 Enter the starting vertex: 1

 The node which are reachable are:
 1       2       3
```

Source Code:

```
#include<stdio.h>
void DFS(int);
int G[10][10],visited[10],n;
void main()
{
   int i,j;    printf("Enter number of vertices:");
   scanf("%d",&n);   printf("\nEnter adjecency matrix of the graph:\n");
   for(i=0; i<n; i++)
   for(j=0; j<n; j++)
   scanf("%d",&G[i][j]);
   for(i=0; i<n; i++) visited[i]=0; DFS(0);
}
 void DFS(int i)
{   int j;
   printf("\n%d",i);
   visited[i]=1;
   for(j=0; j<n; j++)
   if(!visited[j]&&G[i][j]==1)
    DFS(j); }
```

Input and Output:

```
Enter number of vertices: 4
Enter adjacency matrix of the graph:
 1 0 1 1
 1 1 1 1
 0 1 1 0
 1 1 0 1
 The node which are reachable are: 0   2    1     3
```

Source Code:

```
#include<stdio.h>
int ary[10][10],completed[10],n,cost=0;
void takeInput()
{
    int i,j;    printf("Enter the number of node: ");
    scanf("%d",&n);   printf("\nEnter the Cost Matrix\n");
    for(i=0; i < n; i++)
    {
        for( j=0; j < n; j++) scanf("%d",&ary[i][j]);    completed[i]=0;
    }
}
void mincost(int city)
{
    int i,ncity;
       completed[city]=1;     printf("%d--->",city+1);      ncity=least(city);
   if(ncity==999)
    {
        ncity=0;     printf("%d",ncity+1);
        cost+=ary[city][ncity];
        return;
}
    mincost(ncity);
}
int least(int c)
{
    int i,nc=999;
```

```
    int min=999,kmin;
    for(i=0; i < n; i++)
    {      if((ary[c][i]!=0)&&(completed[i]==0))
           if(ary[c][i]+ary[i][c] < min)
           {  min=ary[i][0]+ary[c][i];
              kmin=ary[c][i];
              nc=i;
           }
    }
   if(min!=999) cost+=kmin;
   return nc;
}
int main()
{  takeInput();      printf("\n\nThe Path is:\n");
    mincost(0); //passing 0 because starting vertex printf("\n\nMinimum cost is %d\n ",cost);
    return 0;
}
```

Input and Output:

Enter the number of node: 4

Enter the Cost Matrix:

4 0 2 1

5 1 0 2

3 0 3 1

6 8 1 0

The Path is: 1--->3--->4--->2--->1

Minimum cost is 16

Source Code:

5(a):

```
import pandas as pd
data = {'Name': ['John', 'Mary', 'Peter', 'David'],
'Age': [25, 30, 27, 28],
'Country': ['USA', 'Canada', 'Australia', 'USA']}
df = pd.DataFrame(data)
print(df)
#Load dataset in Python
import pandas as pd
df = pd.read_csv('my_dataset.csv')
print(df)
```

5(b):

```
 import statistics
 # Sample dataset
 data = [2, 4, 6, 2,8, 15]
 # Compute mean
 mean = statistics.mean(data)
 print("Mean: ", mean)
 # Compute median
median = statistics.median(data)
 print("Median: ", median)
 # Compute mode
 mode = statistics.mode(data)
 print("Mode: ", mode)
 # Compute variance
```

```
variance = statistics.variance(data)
print("Variance: ", variance)
# Compute standard deviation
std_deviation = statistics.stdev(data)
print("Standard Deviation: ", std_deviation)
```

Output:

5(a):

	Name	Age	Country
0	John	25	USA
1	Mary	30	Canada
2	Peter	27	Australia
3	David	28	USA

5(b):

```
Mean: 6.167
Median: 5.0
Mode: 2
Variance: 24.167
Standard Deviation: 4.916
```

Source Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('risk.csv')
print(df)
print("\n")
x = df[['speed']]
y = df['risk']
print(x)
print("\n")
print(y)

plt.scatter(df['speed'],df['risk'])
plt.xlabel('Speed of Car')
plt.ylabel('Risk on driving')
plt.title('Car driving speed risk')


from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest =
train_test_split(x,y,test_size=.40,random_state=1)
print(xtrain)
print("\n")
print(xtest)
print("\n")
print(ytrain)
print("\n")
print(ytest)
print("\n")

#60% For training and 40% for test
#xtrain,xtest,ytrain,ytest =
train_test_split(x,y,test_size=.40,random_state=1)
#xtrain
#xtest

#ytrain
#ytest

from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(xtrain,ytrain)

reg.predict(xtest)
print(ytest)

plt.scatter(df['speed'],df['risk'],marker='*',color='red')
plt.xlabel('Speed of Car')
plt.ylabel('Risk on driving')
plt.title('Car driving speed risk')
plt.plot(df.speed,reg.predict(df[['speed']]))
print(reg.predict([[180]]))
regcof = reg.coef_
```

```
print(regcof)
reginter = reg.intercept_
print(reginter)
m = regcof*100+reginter
print(m)
plt.show()
```

Output:

```
   speed  risk
0    200   95
1     90   20
2    300   98
3    110   60
4    240   72
5    115   10
6     50    7
7    230   85
8    190   45
9    260   91
10   290   82
11   185   59
12   310   93
13    95   18
14    30    2
```

```
   speed
0    200
1     90
2    300
3    110
```

4 240

5 115

6 50

7 230

8 190

9 260

10 290

11 185

12 310

13 95

14 30

0 95

1 20

2 98

3 60

4 72

5 10

6 7

7 85

8 45

9 91

10 82

11 59

12 93

13 18

14 2

Name: risk, dtype: int64

	speed
1	90
13	95
0	200
14	30
9	260
8	190
12	310
11	185
5	115

	speed
3	110
7	230
6	50
2	300
10	290
4	240

1	20
13	18
0	95
14	2
9	91
8	45
12	93
11	59

5 10

Name: risk, dtype: int64

3 60

7 85

6 7

2 98

10 82

4 72

Name: risk, dtype: int64

3 60

7 85

6 7

2 98

10 82

4 72

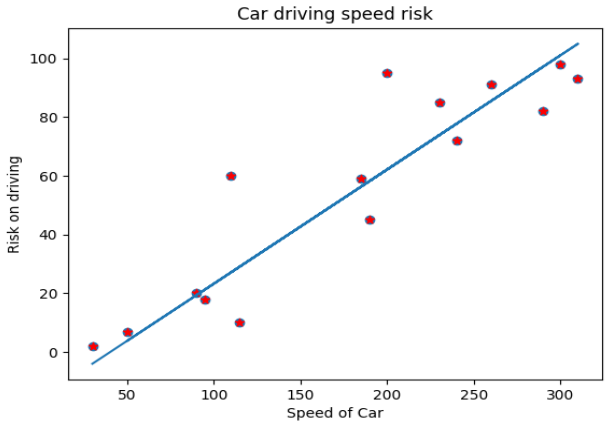
Name: risk, dtype: int64

[54.37693451]

[0.38891318]

-15.62743726501705

[23.26388039]



Plot:

Source Code:

```
import pandas as pd
import numpy as np
data=pd.read_csv('data.csv')
print(data)

# leave the last column
concept=np.array(data)[:,:-1]
# only access the last column
target=np.array(data)[:,-1]

def train(concept,target):
    for i,value in enumerate(target):
        if value.lower()=='yes':
            specific_h=concept[i].copy()
            break
    for i,value in enumerate(concept):
        if target[i].lower()=='yes':
```

```
        for x in range(len(specific_h)):
            if value[x]!=specific_h[x]:
                specific_h[x]='?'
            else:
                pass;
    return specific_h

result=train(concept,target)
print(result)

day=input("Enter 6 word to check:")
day=day.split()
check=True

for i in range(len(result)):
    if result[i]=='?'or result[i]==day[i]:
        check=True;
    else:
        check=False;
        break;
if check:
    print("Enjoy sport")
else:
    print("Not Possible")
```

Input and Output:

```
    sky air temp humidity    wind water forecast enjoy sport
0  sunny    warm   normal  strong  warm     same       yes
1  sunny    warm     high  strong  warm     same       yes
2  rainy    cold     high  strong  warm   change        no
3  sunny    warm     high  strong  cool   change       yes
['sunny' 'warm' '?' 'strong' '?' '?']
Enter 6 word to check:  sunny warm high strong cool change
Enjoy sport
```

Source Code:

```
# Import the Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

# Import some Data from the iris Data Set
iris = datasets.load_iris()

# Take only the first two features of Data.
# To avoid the slicing, Two-Dim Dataset can be used

X = iris.data[:, :2]
y = iris.target

# C is the SVM regularization parameter
C = 1.0

# Create an Instance of SVM and Fit out the data.
# Data is not scaled so as to be able to plot the support vectors
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)

# create a mesh to plot
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
```

```
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

# Plot the data for Proper Visual Representation
plt.subplot(1, 1, 1)

# Predict the result by giving Data to the model
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)
plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')

# Output the Plot
plt.show()
```

Output: