question 1 building a proper w shape

In [158]:

```python
height = 6
for i in range(height):
    if i <= height // 2 or i == height - 1:
        left_spaces = " " * i
        middle_spaces = " " * (2 * (height - i) - 1)
        stars = "*" if i == height -1 else "**"
        print(left_spaces + stars + middle_spaces + stars)
```

```
**           **
 **         **
  **       **
   **     **
     * *
```

In [ ]:

create 8*8 matrix and fill it with checker baord pattern checker board pattern refers to alternating zeros and ones across rows and columns

In [4]:

```python
import numpy as np


matrix = np.ones((8, 8),dtype='i')

matrix[::2, ::2] = 0
matrix[1::2, 1::2] = 0

print(matrix)
```

```
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

In [ ]:

Q3 # for the given data-set of heart diseas draw box plot

In [4]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
dataset=pd.read_csv('heart.csv')
dataset.head()
```

Out[4]:

	Age	Sex	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST
0	40	M	140	289	0	Normal	172	N	0.0	
1	49	F	160	180	0	Normal	156	N	1.0	
2	37	M	130	283	0	ST	98	N	0.0	
3	48	F	138	214	0	Normal	108	Y	1.5	
4	54	M	150	195	0	Normal	122	N	0.0	

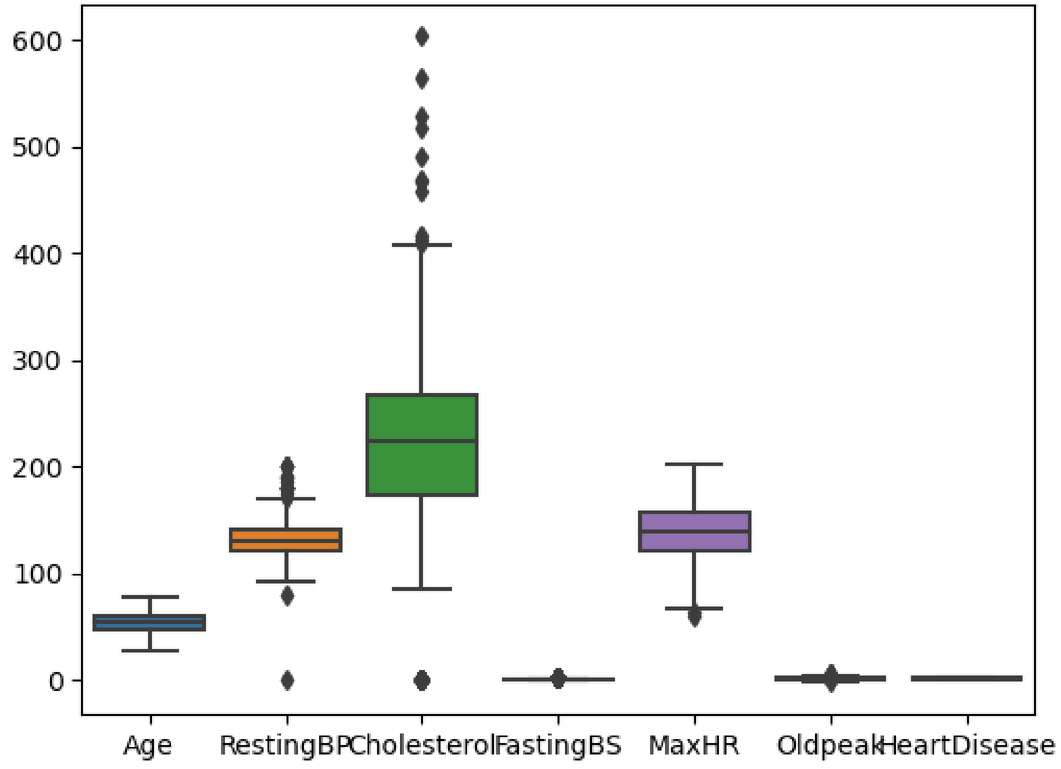
plotting box plot

In [56]:

```
sns.boxplot(data=dataset)
```

Out[56]:

```
<Axes: >
```



Question 4

In [6]:

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler,OneHotEncoder
from sklearn.tree import plot_tree
```

In [30]:

```
dataset=pd.read_csv('heart.csv')
dataset.head()

```

Out[30]:

	Age	Sex	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_
0	40	M	140	289	0	Normal	172	N	0.0	
1	49	F	160	180	0	Normal	156	N	1.0	
2	37	M	130	283	0	ST	98	N	0.0	
3	48	F	138	214	0	Normal	108	Y	1.5	
4	54	M	150	195	0	Normal	122	N	0.0	

In [107]:

```
dataset.isna().sum()
```

Out[107]:

```
Age               0
Sex               0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [20]:

```

```

In [12]:

```
x=dataset.drop('HeartDisease',axis=1)
y=dataset.HeartDisease

```

In [13]:

```
one_hot=OneHotEncoder()
features=['Sex','RestingECG','ExerciseAngina','ST_Slope']
transformer=ColumnTransformer([('one_hot',one_hot,features)],remainder='passthrough')
x_transformed=transformer.fit_transform(x)

```

In [14]:

```
clf=RandomForestClassifier(n_estimators=100)
```

In [21]:

```

x_train,x_test,y_train,y_test=train_test_split(x_transformed,y,test_size=0.2,random_state=
clf.fit(x_train,y_train);
```

model_evaluation

In [22]:

```
clf.score(x_test,y_test)

```

Out[22]:

```
0.9130434782608695
```

Prediction

In [24]:

```
y_preds=clf.predict(x_test)
y_preds
```

Out[24]:

```
array([1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1], dtype=int64)
```

In [25]:

```
predicted_data=pd.DataFrame(y_preds,columns=['prediction'])
actual=pd.DataFrame({'y_test':y_test}).reset_index(drop=True)
pre_vs_actual=pd.concat([actual,predicted_data],axis=1)

pre_vs_actual.head()
```

Out[25]:

	y_test	prediction
0	1	1
1	1	1
2	1	1
3	1	1
4	1	1

In [ ]:

```

```

In [ ]:

```
# confusion_matrix
```

In [105]:

```
cm=confusion_matrix(y_test,y_preds)
cm
```

Out[105]:

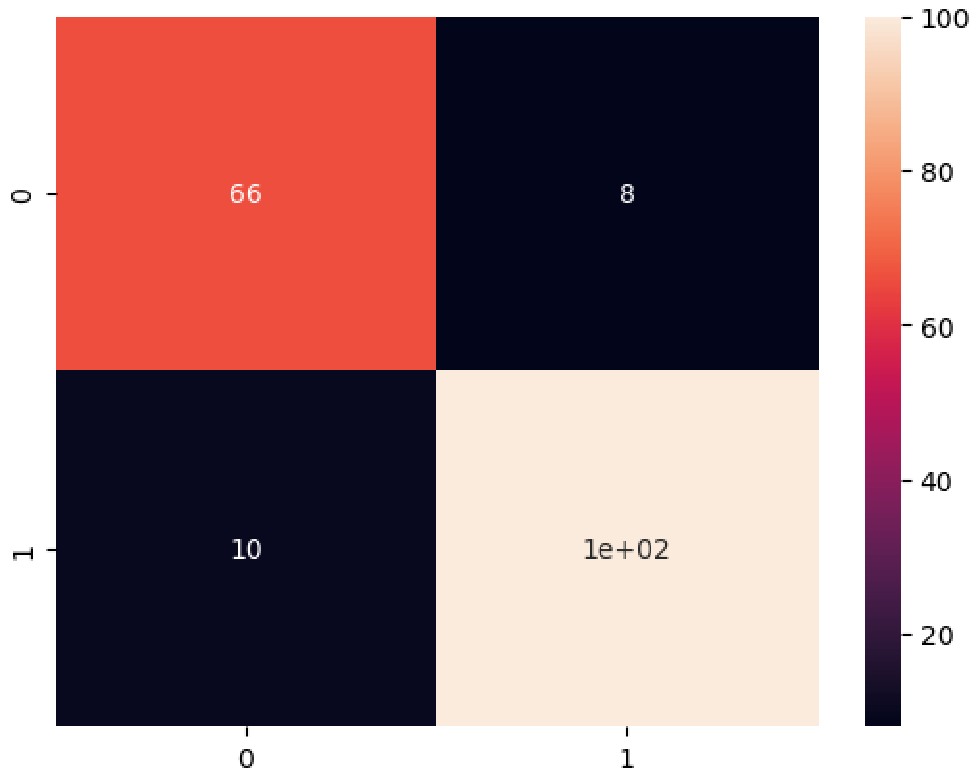
```
array([[ 66,   8],
       [ 10, 100]], dtype=int64)
```

In [106]:

```
sns.heatmap(cm,annot=True)
```

Out[106]:

```
<Axes: >
```



Question 5

Random Forest Algorithm

Select random samples from a given data or training set. This algorithm will construct a decision tree for every training data. Voting will take place by averaging the decision tree. Finally, select the most voted prediction result as the final prediction result. conclusion so instead of depending on one decision tree, the random forest takes the vote or prediction from each tree and based on the majority votes , predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Type *Markdown* and LaTeX: α^2

In [ ]:

```

```