In [78]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [79]:

```
data = pd.read_csv("Heart_Disease_Prediction.csv")
```

Here, Male:1 Female:0
chest pain type (4 values -Ordinal):
Value 1: typical angina,
Value 2: atypical angina,
Value 3: non-anginal pain,
Value 4: asymptomatic
BP:resting Blood Pressure
FBS:Fasting Blood Sugar
EKG:Electrocardiogram Values
Max HR:Max Heart Rate
Exercise angina (binary) (1 = yes; 0 = no)
ST depression:induced by exercise relative to rest
Slope of ST segment:(Ordinal) (Value 1: up sloping , Value 2: flat , Value 3: down sloping)
Fluro:number of major vessels (0–3, Ordinal) observed by fluroscopy
Thallium:maximum heart rate achieved — (Ordinal): 3 = normal; 6 = fixed defect; 7 = reversible defect

In [80]:

```
data.nunique(axis=0)# returns the number of unique values for each variable.
```

Out[80]:

```
Age                         41
Sex                          2
Chest pain type              4
BP                          47
Cholesterol                144
FBS over 120                 2
EKG results                  3
Max HR                      90
Exercise angina              2
ST depression               39
Slope of ST                  3
Number of vessels fluro      4
Thallium                     3
Heart Disease                2
dtype: int64
```

In [81]:

```
data.describe()
```

Out[81]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000

In [82]:

```
print(data.head())
```

```
   Age  Sex  Chest pain type   BP  Cholesterol  FBS over 120  EKG results  
\
0   70    1                4  130          322             0            2   
1   67    0                3  115          564             0            2   
2   57    1                2  124          261             0            0   
3   64    1                4  128          263             0            0   
4   74    0                2  120          269             0            2   

   Max HR  Exercise angina  ST depression  Slope of ST  \
0     109                0            2.4            2   
1     160                0            1.6            2   
2     141                0            0.3            1   
3     105                1            0.2            2   
4     121                1            0.2            1   

   Number of vessels fluro  Thallium Heart Disease  
0                        3         3     Presence  
1                        0         7      Absence  
2                        0         7     Presence  
3                        1         7      Absence  
4                        1         3      Absence  
```

In [83]:

```
print(data.tail())
```

```
     Age  Sex  Chest pain type   BP  Cholesterol  FBS over 120  EKG result
s  \
265   52    1                3  172          199             1
0
266   44    1                2  120          263             0
0
267   56    0                2  140          294             0
2
268   57    1                4  140          192             0
0
269   67    1                4  160          286             0
2

     Max HR  Exercise angina  ST depression  Slope of ST  \
265     162                0            0.5            1
266     173                0            0.0            1
267     153                0            1.3            2
268     148                0            0.4            2
269     108                1            1.5            2

     Number of vessels fluro  Thallium Heart Disease
265                        0         7       Absence
266                        0         7       Absence
267                        0         3       Absence
268                        0         6       Absence
269                        3         3      Presence
```

In [84]:

```
# Display the Missing Values
print(data.isna().sum())
```

```
Age                        0
Sex                        0
Chest pain type            0
BP                         0
Cholesterol                0
FBS over 120               0
EKG results                0
Max HR                     0
Exercise angina            0
ST depression              0
Slope of ST                0
Number of vessels fluro    0
Thallium                   0
Heart Disease              0
dtype: int64
```

In [85]:

```
data.columns
```

Out[85]:

```
Index(['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over 12
0',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart Diseas
e'],
      dtype='object')
```

In [86]:

```
data['Heart Disease'].value_counts()
```

Out[86]:

```
Absence     150
Presence    120
Name: Heart Disease, dtype: int64
```

In [87]:

```
# Separate the features (X) and target (y)
X = data.drop(columns=['Heart Disease'])
Y = data['Heart Disease']
```

In [88]:

```
X
```

Out[88]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	N v
0	70	1	4	130	322	0	2	109	0	2.4	2	
1	67	0	3	115	564	0	2	160	0	1.6	2	
2	57	1	2	124	261	0	0	141	0	0.3	1	
3	64	1	4	128	263	0	0	105	1	0.2	2	
4	74	0	2	120	269	0	2	121	1	0.2	1	
...	...	...	...	...	...	...	...	...	...	...	...	
265	52	1	3	172	199	1	0	162	0	0.5	1	
266	44	1	2	120	263	0	0	173	0	0.0	1	
267	56	0	2	140	294	0	2	153	0	1.3	2	
268	57	1	4	140	192	0	0	148	0	0.4	2	
269	67	1	4	160	286	0	2	108	1	1.5	2	

270 rows × 13 columns

In [89]:

```
Y
```

Out[89]:

```
0      Presence
1       Absence
2      Presence
3       Absence
4       Absence
         ...
265     Absence
266     Absence
267     Absence
268     Absence
269    Presence
Name: Heart Disease, Length: 270, dtype: object
```

In [90]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42
```

In [91]:

```
# Scale the features for better model performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [92]:

```
# Create and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[92]:

```
LogisticRegression()
```

In [93]:

```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

In [94]:

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9074074074074074
```

In [95]:

```
# Generate a classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

     Absence       0.91      0.94      0.93        33
    Presence       0.90      0.86      0.88        21

    accuracy                           0.91        54
   macro avg       0.91      0.90      0.90        54
weighted avg       0.91      0.91      0.91        54
```

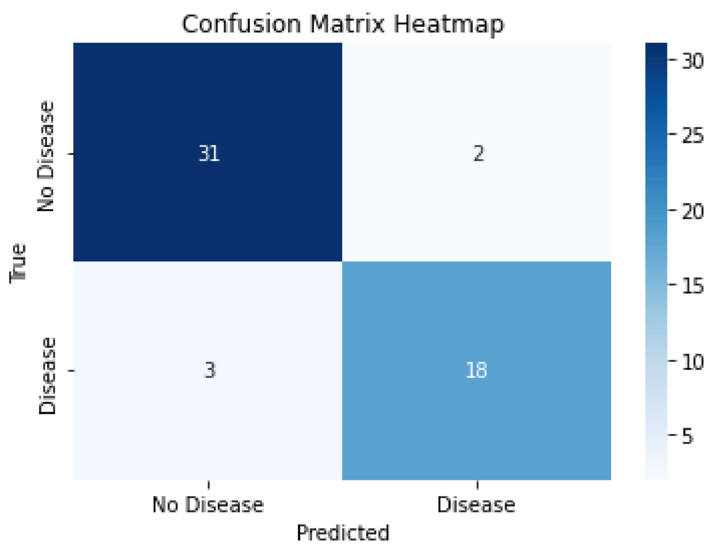
In [96]:

```
# Generate a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[31  2]
 [ 3 18]]
```

In [97]:

```
# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["No Disease", "
            yticklabels=["No Disease", "Disease"])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix Heatmap')
plt.show()
```



True Positive (TP): The number of samples correctly predicted as positive (heart disease patients) - 18

True Negative (TN): The number of samples correctly predicted as negative (non-heart disease patients) - 31

False Positive (FP): The number of samples incorrectly predicted as positive (non-heart disease patients incorrectly classified as heart disease patients) - 2

False Negative (FN): The number of samples incorrectly predicted as negative (heart disease patients

True Positives (TP): There are 18 patients who were correctly predicted as having heart disease.

True Negatives (TN): There are 31 patients who were correctly predicted as not having heart disease.

False Positives (FP): There are 2 patients who were incorrectly predicted as having heart disease, but they actually don't have heart disease.

False Negatives (FN): There are 3 patients who were incorrectly predicted as not having heart disease, but they actually do have heart disease.

Accuracy: (TP + TN) / Total = (18 + 31) / (18 + 31 + 2 + 3) ≈ 0.91 or 91%. Accuracy represents the overall correctness of the model's predictions.

Precision: TP / (TP + FP) = 18 / (18 + 2) ≈ 0.90 or 90%. Precision tells us the proportion of positive predictions that were correct.

Recall (Sensitivity): TP / (TP + FN) = 18 / (18 + 3) ≈ 0.86 or 86%. Recall indicates the proportion of actual positives that were correctly identified.

Specificity: TN / (TN + FP) = 31 / (31 + 2) ≈ 0.94 or 94%. Specificity tells us the proportion of actual negatives that were correctly identified.

F1 Score: 2 (Precision Recall) / (Precision + Recall) ≈ 0.88 or 88%. F1 score is the harmonic mean of precision and recall, giving us a single metric to consider both.

In [ ]:

```

```