In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
```

In [2]:
```
file= pd.read_csv(r"C:\Users\HP\Desktop\logistic regression.csv")
print(file)
```

```
           x1         x2  y
0   34.623660  78.024693   0
1   30.286711  43.894998   0
2   35.847409  72.902198   0
3   60.182599  86.308552   1
4   79.032736  75.344376   1
..        ...        ...  ..
95  83.489163  48.380286   1
96  42.261701  87.103851   1
97  99.315009  68.775409   1
98  55.340018  64.931938   1
99  74.775893  89.529813   1

[100 rows x 3 columns]
```

In [3]:
```
x=file.iloc[:,[0,1]].values
print(x)
```

```
[[34.62365962 78.02469282]
 [30.28671077 43.89499752]
 [35.84740877 72.90219803]
 [60.18259939 86.3085521 ]
 [79.03273605 75.34437644]
 [45.08327748 56.31637178]
 [61.10666454 96.51142588]
 [75.02474557 46.55401354]
 [76.0987867  87.42056972]
 [84.43281996 43.53339331]
 [95.86155507 38.22527806]
 [75.01365839 30.60326323]
 [82.30705337 76.4819633 ]
 [69.36458876 97.71869196]
 [39.53833914 76.03681085]
 [53.97105215 89.20735014]
 [69.07014406 52.74046973]
 [67.94685548 46.67857411]
 [70.66150955 92.92713789]
 [76.97878373 47.57596365]
 [67.37202755 42.83843832]
 [89.67677575 65.79936593]
 [50.53478829 48.85581153]
 [34.21206098 44.2095286 ]
 [77.92409145 68.97235999]
 [62.27101367 69.95445795]
 [80.19018075 44.82162893]
 [93.1143888  38.80067034]
 [61.83020602 50.25610789]
 [38.7858038  64.99568096]
 [61.37928945 72.80788731]
 [85.40451939 57.05198398]
 [52.10797973 63.12762377]
 [52.04540477 69.43286012]
 [40.23689374 71.16774802]
 [54.63510555 52.21388588]
 [33.91550011 98.86943574]
```

```
[64.17698887 80.90806059]
[74.78925296 41.57341523]
[34.18364003 75.23772034]
[83.90239366 56.30804622]
[51.54772027 46.85629026]
[94.44336777 65.56892161]
[82.36875376 40.61825516]
[51.04775177 45.82270146]
[62.22267576 52.06099195]
[77.19303493 70.4582    ]
[97.77159928 86.72782233]
[62.0730638  96.76882412]
[91.5649745  88.69629255]
[79.94481794 74.16311935]
[99.27252693 60.999031  ]
[90.54671411 43.39060181]
[34.52451385 60.39634246]
[50.28649612 49.80453881]
[49.58667722 59.80895099]
[97.64563396 68.86157272]
[32.57720017 95.59854761]
[74.24869137 69.82457123]
[71.79646206 78.45356225]
[75.39561147 85.75993667]
[35.28611282 47.02051395]
[56.2538175  39.26147251]
[30.05882245 49.59297387]
[44.66826172 66.45008615]
[66.56089447 41.09209808]
[40.45755098 97.53518549]
[49.07256322 51.88321182]
[80.27957401 92.11606081]
[66.74671857 60.99139403]
[32.72283304 43.30717306]
[64.03932042 78.03168802]
[72.34649423 96.22759297]
[60.45788574 73.0949981 ]
[58.84095622 75.85844831]
[99.8278578  72.36925193]
[47.26426911 88.475865  ]
[50.4581598  75.80985953]
[60.45555629 42.50840944]
[82.22666158 42.71987854]
[88.91389642 69.8037889 ]
[94.83450672 45.6943068 ]
[67.31925747 66.58935318]
[57.23870632 59.51428198]
[80.366756   90.9601479 ]
[68.46852179 85.5943071 ]
[42.07545454 78.844786  ]
[75.47770201 90.424539  ]
[78.63542435 96.64742717]
[52.34800399 60.76950526]
[94.09433113 77.15910509]
[90.44855097 87.50879176]
[55.48216114 35.57070347]
[74.49269242 84.84513685]
[89.84580671 45.35828361]
[83.48916274 48.3802858 ]
[42.26170081 87.10385094]
[99.31500881 68.77540947]
[55.34001756 64.93193801]
[74.775893   89.5298129 ]]
```

In [4]:

```
y=file.iloc[:,2].values
print(y)
```

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

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

In [5]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain, ytest = train_test_split(x,y,test_size=0.25,random_state=0)
```

In [6]:
```
from sklearn.linear_model import LogisticRegression
classifier =  LogisticRegression(random_state=0)
classifier.fit(xtrain,ytrain)
```

Out[6]:
```
LogisticRegression(random_state=0)
```

In [7]:
```
y_pred = classifier.predict(xtest)
```

In [8]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(ytest,y_pred)
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[11  0]
 [ 4 10]]
```

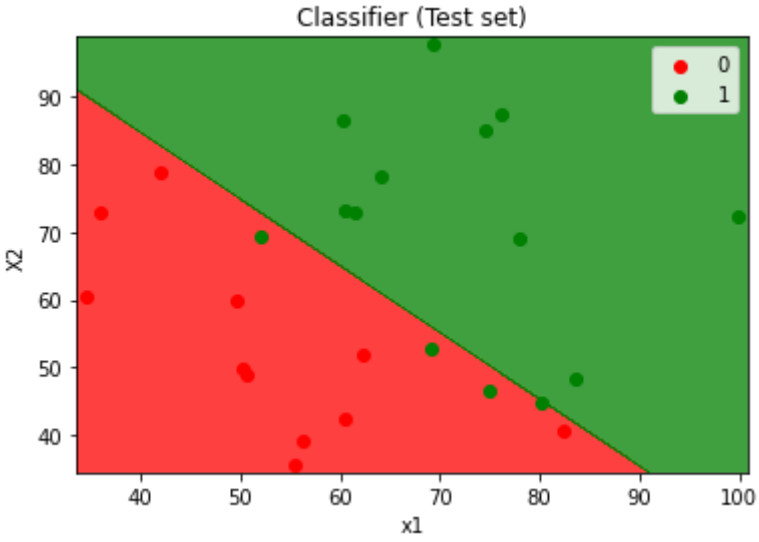
In [9]:
```
from sklearn.metrics import accuracy_score
print("accuracy:",accuracy_score(ytest,y_pred))
```

```
accuracy: 0.84
```

In [11]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = xtest, ytest
X1, X2 = np.meshgrid(np.arange (start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(
np.array([X1.ravel(), X2.ravel()]).T).reshape(
X1. shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifier (Test set)')
plt.xlabel('x1')
plt.ylabel('X2')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoid
ed as value-mapping will have precedence in case its length matches with *x* & *y*.
Please use the *color* keyword-argument or provide a 2-D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoid
ed as value-mapping will have precedence in case its length matches with *x* & *y*.
Please use the *color* keyword-argument or provide a 2-D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
```


Classifier (Test set)
0
1
X2
x1
90
80
70
60
50
40
40
50
60
70
80
90
100

In [ ]: