According to the above results, when k is equal to 25 and 30 to the KNN classifier, the accuracy is the highest. But based on speed of convergence and generalisation I pick 30 to be the best parameter. I set up the K to be 30 because of the accuracy. The lower value of k will predict a more locally model, and higher value of k will predict a more globally model. The smaller K can make the model more complex and consume too much to the computer, it is not good to the complexity(big O) and speed of convergence. It may also have overfitting problem.Its accuracy is 100% on the test set.The accuracy of the model is 100% on the test set when K=30.

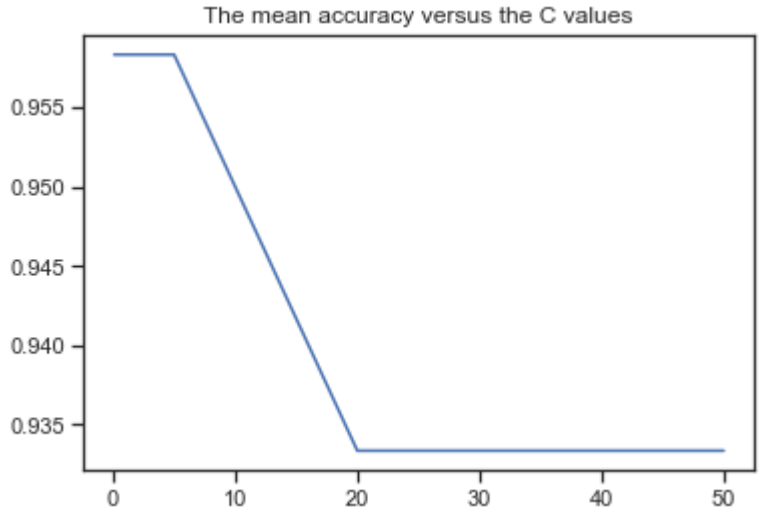
Question 3: SVM

In [36]:
```
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score

# First, randomly divide data into (80%, 20%) portions of train-validation and test set
s (with random state=42).
X_train1, X_test1, y_train1, y_test1= train_test_split(data.data,data.target, test_size
=0.2, random_state=42)

# apply 10-fold cross validation on the train-validation set. In every fold, 90% of dat
a is used for training and 10% of
#data for validation. For every C value, a mean accuracy of the folds is found.
C_array=[0.1, 0.5, 1, 2, 5, 10, 20, 50]

cache1=[] #cache to store the result
for c in C_array:
    clf_svm = SVC(kernel='linear', C=c)
    scores = cross_val_score(clf_svm, X_train1, y_train1, cv=10)
    cache1.append(np.mean(scores))

    print(scores)

cache1
```

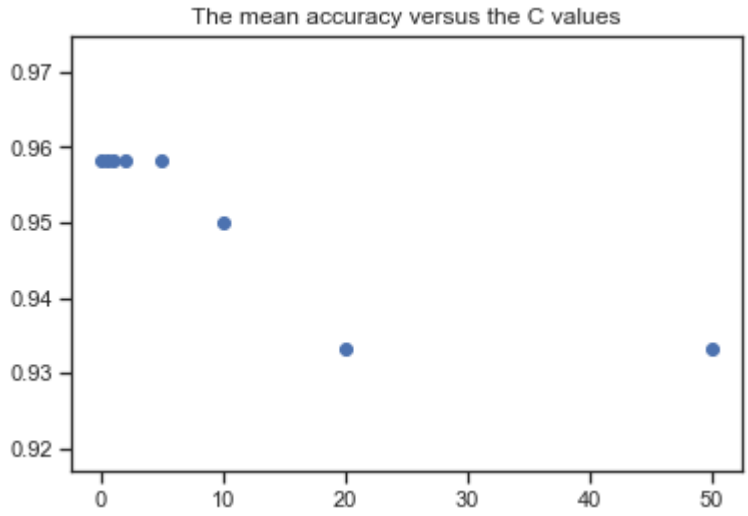
```
[1.         1.         1.         1.         0.83333333 0.83333333
 1.         1.         1.         0.91666667]
[1.         1.         0.91666667 1.         0.91666667 0.83333333
 1.         1.         1.         0.91666667]
[1.         1.         0.91666667 1.         0.91666667 0.83333333
 1.         1.         1.         0.91666667]
[1.         1.         0.91666667 1.         0.83333333 0.83333333
 1.         1.         1.         1.        ]
[1.         1.         0.91666667 1.         0.83333333 0.83333333
 1.         1.         1.         1.        ]
[1.         1.         0.91666667 1.         0.83333333 0.83333333
 0.91666667 1.         1.         1.        ]
[0.91666667 1.         0.91666667 1.         0.75       0.83333333
 0.91666667 1.         1.         1.        ]
[0.91666667 1.         0.91666667 1.         0.75       0.83333333
 1.         0.91666667 1.         1.        ]
```

Out[36]:
```
[0.9583333333333334,
 0.9583333333333333,
 0.9583333333333333,
 0.9583333333333334,
 0.9583333333333334,
 0.95,
 0.9333333333333332,
 0.9333333333333332]
```

In [37]:
```
#Plot the mean accuracy versus the C values.
plt.title('The mean accuracy versus the C values')
plt.plot(C_array,cache1)
plt.show()
```



In [38]:
```
plt.title('The mean accuracy versus the C values')
plt.scatter(C_array,cache1)
plt.show()
```



According to the two figures, not only one parameter has the highest accuracy. C=5 the best C parameter for the SVM classifier. I set up the maximum depth to be 5 based on speed of convergence and generalisation The greater of C, the less tolerance for error.I also tried C=2,but it may also have overfitting problem. But if C is too small to trend to zero, it may cause underfitting problem.

In [39]:
```
#Train the model using the train-validation set.
clf_svm.fit(X_train1, y_train1)
```

Out[39]:
```
SVC(C=50, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [40]:
```
# Finally, report the test accuracy.
clf_test= SVC(kernel='linear', C=5)
clf_test.fit(X_train1, y_train1)
scores_test=clf_test.score(X_test1, y_test1)
print(scores_test)
```

```
0.9666666666666667
```

So the test accuracy is 0.9666666666666667 on the test set when C is 5.It is the best C parameter for the SVM classifier. I set up the maximum depth to be 5 based on speed of convergence and generalisation The greater of C, the less tolerance for error.I also tried C=2,but it may also have overfitting problem duo to the test score is 100%. The model's accuracy is 0.9666666666666667 on the test set when C=5.

Question 4: Tree-based Classifiers