In [58]:

```
#Import scikit libraries
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

#Import numpy library
import numpy as np

#Import pandas library
import pandas as pd
```

In [10]:

```
german = pd.read_csv('/Users/juanrquilesjr/Documents/Machine_Learning-Spring_2019/assignment-5/german.data-numeric.csv')
german.head()
```

Out[10]:

	A-1	A-2	A-3	A-4	A-5	A-6	A-7	A-8	A-9	A-10	...	A-16	A-17	A-18	A-19	A-20	A-21	A-22	A-23	A-24	C
0	1	6	4	12	5	5	3	4	1	67	...	0	0	1	0	0	1	0	0	1	Good
1	2	48	2	60	1	3	2	2	1	22	...	0	0	1	0	0	1	0	0	1	Bad
2	4	12	4	21	1	4	3	3	1	49	...	0	0	1	0	0	1	0	1	0	Good
3	1	42	2	79	1	4	3	4	2	45	...	0	0	0	0	0	0	0	0	1	Good
4	1	24	3	49	1	3	3	4	4	53	...	1	0	1	0	0	0	0	0	1	Bad

5 rows × 25 columns

In [23]:

```
data = german.drop(['C'], axis = 1)
data.head()
X = data
```

In [25]:

```
y = german.C
```

In [34]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state =4)

knn = KNeighborsClassifier(n_neighbors = 1)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print(y_pred)
```

```
['Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Good'
 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Bad'
 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good'
 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good'
 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Bad' 'Bad'
 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Bad' 'Bad' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Bad' 'Bad'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Bad' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good'
 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good'
 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good'
 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Bad' 'Bad' 'Bad' 'Good'
 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Good'
 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good']
```

In [44]:

```
from sklearn.cross_validation import cross_val_score

print(cross_val_score(knn, X, y, cv = 5, scoring = 'f1_weighted').mean())
```

```
0.6492631890597527
```

In [49]:

```
# k range from 1 thru 5
k_range = range(1,11)

f1_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    scores = cross_val_score(knn, X, y, cv = 5, scoring = 'f1_weighted')
    f1_scores.append(scores.mean())

print(f1_scores)
```

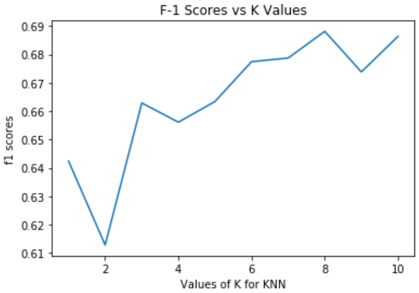
```
[0.6492631890597527, 0.6157121158061004, 0.6791265101847322, 0.6711186458775723, 0.6732563362205534, 0.6765344633226273, 0.683349672523517, 0.6946
521510217428, 0.6623751521389529, 0.6863855863776278]
```

In [110]:

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(k_range, f1_scores)
plt.title('F-1 Scores vs K Values')
plt.xlabel('Values of K for KNN')
plt.ylabel('f1 scores')
```

Out[110]:

```
Text(0, 0.5, 'f1 scores')
```



In [100]:

```
from sklearn.feature_selection import SelectKBest, chi2

best = SelectKBest(chi2, k = 10).fit_transform(X,y)
```

In [101]:

```
print(best)
```

```
[[ 1  6  4 ...  0  0  0]
 [ 2 48  2 ...  0  0  0]
 [ 4 12  4 ...  0  0  0]
 ...
 [ 4 12  2 ...  0  0  0]
 [ 1 45  2 ...  0  0  0]
 [ 2 45  4 ...  0  1  0]]
```

In [102]:

```
df_best = pd.DataFrame(best)
df_best.head(10)
```

Out[102]:

	0	1	2	3	4	5	6	7	8	9
0	1	6	4	12	5	1	67	0	0	0
1	2	48	2	60	1	1	22	0	0	0
2	4	12	4	21	1	1	49	0	0	0
3	1	42	2	79	1	2	45	0	0	0
4	1	24	3	49	1	4	53	1	0	0
5	4	36	2	91	5	4	35	0	0	0
6	4	24	2	28	3	2	53	0	0	0
7	2	36	2	69	1	3	35	0	1	1
8	4	12	2	31	4	1	61	0	0	0
9	2	30	4	52	1	3	28	1	0	0

In [103]:

```
german.head(10)
```

Out[103]:

	A-1	A-2	A-3	A-4	A-5	A-6	A-7	A-8	A-9	A-10	...	A-16	A-17	A-18	A-19	A-20	A-21	A-22	A-23	A-24	C
0	1	6	4	12	5	5	3	4	1	67	...	0	0	1	0	0	1	0	0	1	Good
1	2	48	2	60	1	3	2	2	1	22	...	0	0	1	0	0	1	0	0	1	Bad
2	4	12	4	21	1	4	3	3	1	49	...	0	0	1	0	0	1	0	1	0	Good
3	1	42	2	79	1	4	3	4	2	45	...	0	0	0	0	0	0	0	0	1	Good
4	1	24	3	49	1	3	3	4	4	53	...	1	0	1	0	0	0	0	0	1	Bad
5	4	36	2	91	5	3	3	4	4	35	...	0	0	1	0	0	0	0	1	0	Good
6	4	24	2	28	3	5	3	4	2	53	...	0	0	1	0	0	1	0	0	1	Good
7	2	36	2	69	1	3	3	2	3	35	...	0	1	1	0	1	0	0	0	0	Good
8	4	12	2	31	4	4	1	4	1	61	...	0	0	1	0	0	1	0	1	0	Good
9	2	30	4	52	1	1	4	2	3	28	...	1	0	1	0	0	1	0	0	0	Bad

10 rows × 25 columns

In [104]:

```
german_best = german[['A-1','A-2','A-3','A-4','A-5','A-9','A-10','A-16', 'A-17', 'A-20']]
german_best
X1 = german_best
X1.shape
```

Out[104]:

```
(959, 10)
```

In [105]:

```
y.shape
```

Out[105]:

```
(959,)
```

```
In [106]: X_train, X_test, y_train, y_test = train_test_split(X1, y, test_size=0.4, random_state =4)

          knn = KNeighborsClassifier(n_neighbors = 1)

          knn.fit(X_train, y_train)

          y_pred = knn.predict(X_test)

          print(y_pred)
```

```
['Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Good'
 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good'
 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Bad'
 'Bad' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Bad' 'Bad' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad'
 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good'
 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Bad' 'Good' 'Bad' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Bad' 'Bad' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Bad' 'Bad'
 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good'
 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad'
 'Bad' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Bad'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Bad'
 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good'
 'Bad' 'Bad' 'Bad' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Good' 'Good'
 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Good' 'Bad' 'Bad'
 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Bad' 'Bad' 'Good' 'Good' 'Good' 'Good'
 'Bad' 'Good' 'Good' 'Bad' 'Good' 'Bad' 'Good' 'Good']
```

```
In [107]: print(cross_val_score(knn, X1, y, cv = 5, scoring = 'f1_weighted').mean())
```

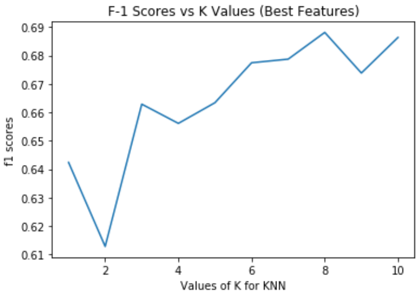
```
0.6423728526515989
```

```
In [115]: # k range from 1 thru 5
          k_range = range(1,11)

          f1_scores = []

          for k in k_range:
              knn = KNeighborsClassifier(n_neighbors = k)
              scores = cross_val_score(knn, X1, y, cv = 5, scoring = 'f1_weighted')
              f1_scores.append(scores.mean())

          print(f1_scores)
```

```
[0.6423728526515989, 0.6127833037347586, 0.6628784274125779, 0.656105519245106, 0.6633922118074406, 0.6774488774388443, 0.678720138652704, 0.68813
46058035991, 0.673829644209499, 0.6863546165207879]
```

```
In [113]: plt.plot(k_range, f1_scores)
          plt.title('F-1 Scores vs K Values (Best Features)')
          plt.xlabel('Values of K for KNN')
          plt.ylabel('f1 scores')
```

```
Out[113]: Text(0, 0.5, 'f1 scores')
```


F-1 Scores vs K Values (Best Features)
f1 scores
Values of K for KNN