In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from hash_table import LinearProbePotionTable
import random

random.seed(0)
size = 100
good_hash_tbl = LinearProbePotionTable(size, True)
bad_hash_tbl = LinearProbePotionTable(size, False)

# To collect statistics of hash tables
badh_conflicts = []
badh_probes = []
badh_prob_max = []
goodh_conflicts = []
goodh_probes = []
goodh_prob_max = []
```

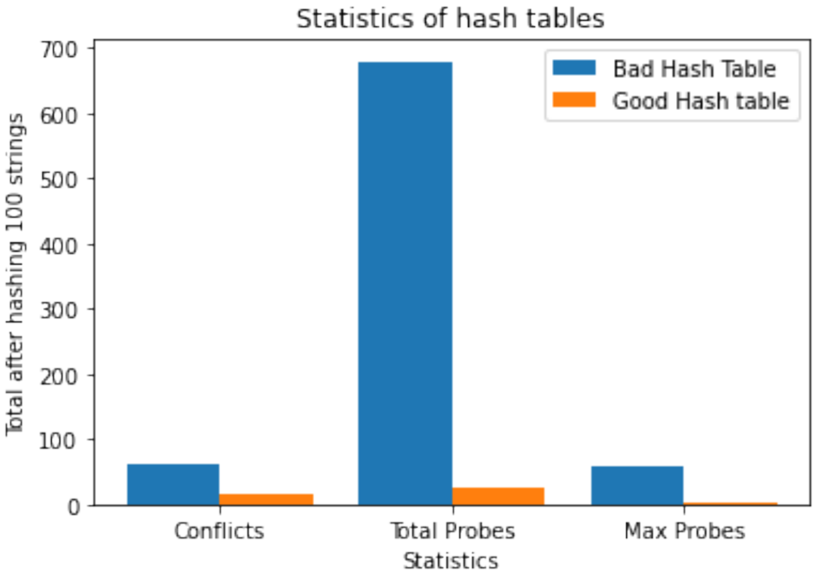
In [2]:

```
for _ in range(0,size):
    string = ""
    rand_length = random.randint(0,20)
    for _ in range(rand_length):
        string += chr(random.randrange(97, 97 + 26))
    # for bad hash
    bad_hash_tbl[string] = string
    badh_conflicts.append(bad_hash_tbl.statistics()[0])
    badh_probes.append(bad_hash_tbl.statistics()[1])
    badh_prob_max.append(bad_hash_tbl.statistics()[2])
    # for good hash
    good_hash_tbl[string] = string
    goodh_conflicts.append(good_hash_tbl.statistics()[0])
    goodh_probes.append(good_hash_tbl.statistics()[1])
    goodh_prob_max.append(good_hash_tbl.statistics()[2])

print("Bad Hash Table Conflicts \n" + str(badh_conflicts))
print("Bad Hash Table Total Probes \n" + str(badh_probes))
print("Bad Hash Table Max Probes \n" + str(badh_prob_max))

print("Good Hash Table Conflicts \n" + str(goodh_conflicts))
print("Good Hash Table Total Probes \n" + str(goodh_probes))
print("Good Hash Table Max Probes \n" + str(goodh_prob_max))
```

```
Bad Hash Table Conflicts
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6,
6, 7, 7, 7, 7, 7, 8, 8, 9, 10, 11, 11, 11, 11, 12, 12, 12, 13, 13, 13, 14,
14, 14, 15, 16, 17, 18, 19, 19, 20, 20, 21, 22, 23, 24, 25, 26, 27, 27, 28,
29, 30, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]
Bad Hash Table Total Probes
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 8,
8, 9, 9, 9, 9, 9, 12, 12, 13, 14, 15, 15, 15, 15, 17, 17, 17, 25, 25, 25, 2
6, 26, 26, 27, 32, 35, 57, 58, 58, 60, 60, 63, 68, 69, 70, 71, 80, 82, 82,
87, 98, 118, 118, 145, 150, 153, 159, 160, 162, 164, 165, 167, 168, 192, 19
3, 197, 220, 253, 258, 260, 291, 339, 347, 365, 398, 449, 495, 541, 542, 54
4, 560, 620, 638, 650, 679]
Bad Hash Table Max Probes
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 8, 8, 8, 8, 8, 8, 8, 8,
8, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 33, 33, 33, 33, 48, 48,
48, 48, 51, 51, 51, 51, 51, 51, 60, 60, 60, 60]
Good Hash Table Conflicts
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 6, 6, 7, 8, 8, 8, 9, 10, 11, 11, 11, 12, 12, 12, 12, 13, 14, 14, 14, 14,
14, 15, 15, 16]
Good Hash Table Total Probes
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 7, 7, 8, 9, 9, 9, 10, 11, 14, 14, 14, 15, 15, 15, 15, 18, 19, 19, 19, 19
, 19, 21, 21, 25]
Good Hash Table Max Probes
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4]
```

In [3]:

```
X = ['Conflicts', 'Total Probes','Max Probes']

BadHashStats = [badh_conflicts[-1], badh_probes[-1], badh_prob_max[-1]]
GoodHashStats = [goodh_conflicts[-1], goodh_probes[-1], goodh_prob_max[-1]

X_axis = np.arange(len(X))

plt.bar(X_axis - 0.2, BadHashStats, 0.4, label = 'Bad Hash Table')
plt.bar(X_axis + 0.2, GoodHashStats, 0.4, label = 'Good Hash table')

plt.xticks(X_axis, X)
plt.xlabel("Statistics")
plt.ylabel("Total after hashing 100 strings")
plt.title("Statistics of hash tables")
plt.legend()
plt.show()

# Explanation:

# As observed from the graph, bad hash table has far more conflicts
# and probes than the good hash table. First of all, there would be
# lesser conflicts as, a good hash's base would change the base every
# hash making it randomized so rate of collision is lower when compared
# to the bad hash function which only considers the first char. Probing
# only occurs when there is a conflict, therefore it will also be lesser.
# Overall the number of probes done by bad hash table would be more than
# good hash table. Hence, the statistics show that all final probes and
# conflicts are more than good hash table after using an example of
# hashing 100 random strings into the hash table.
```


Statistics of hash tables
Bad Hash Table
Good Hash table
Total after hashing 100 strings
Conflicts
Total Probes
Max Probes
Statistics