1. // CPP program to implement double hashing
2. #include <bits/stdc++.h>
3. using namespace std;
5. // Hash table size
6. #define TABLE\_SIZE 13
8. // Used in second hash function.
9. #define PRIME 7
11. class DoubleHash
12. {
13. // Pointer to an array containing buckets
14. int \*hashTable;
15. int curr\_size;
17. public:
19. // function to check if hash table is full
20. bool isFull()
21. {
23. // if hash size reaches maximum size
24. return (curr\_size == TABLE\_SIZE);
25. }
27. // function to calculate first hash
28. int hash1(int key)
29. {
30. return (key % TABLE\_SIZE);
31. }
33. // function to calculate second hash
34. int hash2(int key)
35. {
36. return (PRIME - (key % PRIME));
37. }
39. DoubleHash()
40. {
41. hashTable = new int[TABLE\_SIZE];
42. curr\_size = 0;
43. for (int i=0; i<TABLE\_SIZE; i++)
44. hashTable[i] = -1;
45. }
47. // function to insert key into hash table
48. void insertHash(int key)
49. {
50. // if hash table is full
51. if (isFull())
52. return;
54. // get index from first hash
55. int index = hash1(key);
57. // if collision occurs
58. if (hashTable[index] != -1)
59. {
60. // get index2 from second hash
61. int index2 = hash2(key);
62. int i = 1;
63. while (1)
64. {
65. // get newIndex
66. int newIndex = (index + i \* index2) %
67. TABLE\_SIZE;
69. // if no collision occurs, store
70. // the key
71. if (hashTable[newIndex] == -1)
72. {
73. hashTable[newIndex] = key;
74. break;
75. }
76. i++;
77. }
78. }
80. // if no collision occurs
81. else
82. hashTable[index] = key;
83. curr\_size++;
84. }
86. // function to display the hash table
87. void displayHash()
88. {
89. for (int i = 0; i < TABLE\_SIZE; i++)
90. {
91. if (hashTable[i] != -1)
92. cout << i << " --> "
93. << hashTable[i] << endl;
94. else
95. cout << i << endl;
96. }
97. }
98. };
100. // Driver's code
101. int main()
102. {
103. int a[] = {19, 27, 36, 10, 64};
104. int n = sizeof(a)/sizeof(a[0]);
106. // inserting keys into hash table
107. DoubleHash h;
108. for (int i = 0; i < n; i++)
109. h.insertHash(a[i]);
111. // display the hash Table
112. h.displayHash();
113. return 0;
114. }