Week 1 Challenge Problem
Graded Quiz • 30 min

Due Oct 28, 2:59 PM CST

Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE
100%

Week 1 Challenge Problem

LATEST SUBMISSION GRADE
100%

1. Complete the code below by implementing the function

5 / 5 points

```
int insert(int value, std::vector<int> &table);
```

This insert function should compute a good hash function of *value*. This hash function should return the least-significant three decimal digits (a number from 0 to 999) of the variable *value*. This hash should be used as an index into the thousand-element vector *table* that has been initialized with -1 in each element. If the element at this location of *table* is available (currently set to -1), you can replace the element with *value*. If this location is not available (currently set to some other value than -1) then you should check the next element, repeatedly, until you find an available element and can store *value* there. The insert() function should then return the number of times a location in the hash table was identified to store *value* but was not available.

The main() procedure below will create 500 random values and call insert() on each one of them to insert them into the table. At the end, this procedure will report the length of the longest cluster encountered when inserting a value (as reported by your insert() function) and then print out the contents of the hash table so you can see how clusters form. Since the original hashed position will be the three least significant digits of the value stored there, it will be easy to see which values had to be relocated by linear probing, and how much probing was needed.

When you submit your code, the length of the longest cluster encountered when inserting a value as reported by your insert() function will be compared to the result from the reference code for correctness.

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>
#include <functional>

int insert(int value, std::vector<int> &table) {
  // Code to insert value into a hashed location in table
  // where table is a vector of length 1000.
  // Returns the number of collisions encountered when
  // trying to insert value into table.
  int hash = value % 1000;
  int collisions = 0;

  if( -1 == table[hash] )
  {
    // hash slot is available, update hash table with value
    table[hash] = value;
  }
  else
  {

    while( -1 != table[hash] )
    {
      // hash collision occurrs
      collisions++;

      // linear probe
      hash++;
    }

    // probe a available slot, upte hash table with value
    table[hash] = value;

  }

  return collisions;
}
//end of function insert

int main() {
  int i, j, hit, max_hit = 0, max_value = -1;

  std::vector<int> value(500);

  int old_value = 0;
  for (i = 0; i < 500; i++) {
    old_value += rand()%100;
    value[i] = old_value;
  }

  // create hash table of size 1000 initialized with -1
  std::vector<int> table(1000,-1);

  for (i = 0; i < 500; i++) {
    hit = insert(value[i],table);
    if (hit > max_hit) {
      max_hit = hit;
      max_value = value[i];
    }
  }

  std::cout << "Hashing value " << max_value << " experienced " << max_hit << "
    collisions." << std::endl <<std::endl;

  for (j = 0; j < 1000; j += 10) {
    std::cout << std::setw(3) << j << ":";
    for (i = 0; i < 10; i++) {
      if (table[j+i] == -1)
        std::cout << "      ";
      else
        std::cout << std::setw(6) << table[j+i];
    }
    std::cout << std::endl;
  }
```

```
   }

   return 0;
}
```

Reset

Correct

You counted a maximum of 12 collisions when inserting a single new item into the hash table.