**Software Test Plan**

**For**

**Programming Assignment 4**

**November 11th, 2015**

Prepared By

Poornima Byre Gowda

Prepared for

Dr. Coleman, Instructor

CS221, Data Structures in C++

Computer Science Department

University of Alabama in Huntsville

**Table of Contents**

1.0 System Overview ………………………………………….1

2.0 Referenced Documents…………………………………1

3.0 Test Procedures ……………………………………........1

**Appendices**

A. Test Plan Forms……………………………………………..9

**Software Test Plan**

**1.0 System Overview**

A hash function is the technique used to calculate a unique index into the hash table from a key. This may be a mathematical calculation based on some value of the key or it may be some data manipulation technique. The purpose of this program study the effects of different hash functions on the efficiency of inserting data into a hash table.

**2.0 Referenced Documents**

Statement of work provided for programming assignment 4.

* Lecture slides provided in CS 221.
* Harvey Deitel and Paul Deitel, ““*C++ how to program*”, 8th edition

**3.0 Test Procedures**

**3.1 Source File: main.cpp and Hashing.h**

**3.1.1 Function: 1. void InitTable (HashStruct hashT[], int TableSize)**

**3.1.1.1 Purpose and Procedure**

The Purpose of this function to initialize Hashing table to empty.

**3.1.1.2 Inputs**

1:HashStruct hashT[] -> starting address of the array of Hash Table structure.

2: TableSize -> integer variable which contain the size of the table.

**3.1.1.3 Expected Output**

Intializes the Data in the hash Table to zero.

**3.1.1.4 Success Criteria**

Successfully initialize the value of the Data to zero.

**3.2.2 Function: int HashInsert (HashStruct T[], char \*key, int data, int hNum, int dhNum);**

**3.2.2.1 Purpose and Procedure**

HashInsert shall call one of the Hash functions and set the probe increment to the return value from one of the functions.

**3.2.2.2 Inputs**

HashStruct T[]-> Hash table array of 100 data structures.

Key -> character value key value of the Hash index.

Data->integervalue indicates the Data to be stored in the Hash Table with respective Key value.

hNum -> Indicates the particular Hashing Function.

dhNum-> Indicates the particular Double Hashing functions.

**3.2.2.3 Expected Output**  
The return value of this function shall be a integer count of the number of collisions encountered for the particular key.

**3.2.2.4 Success Criteria**   
Successfully return the number of collisions encountered for this key.

**3.2.3 Function: Hash\_1 (char \*key)**

**3.2.3.1 Purpose and Procedure**

The purpose of this function is to find index value to insert the Data to that respective index value. (finding the index by **Average** method).

**3.2.3.2 Inputs**

Key-> The character Key value

**3.2.3.3 Expected Output**.

Returnthe integer value indicating where the data has to be indexed in the hash Table.  
**3.2.3.4 Success Criteria**   
successfully find the index value.

**3.2.4 Function:** **Hash\_2 (char \*key)**

**3.2.4.1 Purpose and Procedure**

The purpose of this function is to find index value to insert the Data to that respective index value. (finding the **Middle Squaring** method)

**3.2.4.2 Inputs**

Key-> The character Key value

**3.2.4.3 Expected Output**.

Returnthe integer value indicating where the data has to be indexed in the hash Table.  
**3.2.4.4 Success Criteria**   
successfully find the index value.

**3.2.5 Function: Hash\_3 (char \*key)**

**3.2.4.1 Purpose and Procedure**

The purpose of this function is to find index value to insert the Data to that respective index value. (Finding the **Truncation** method)

**3.2.4.2 Inputs**

Key-> The character Key value

**3.2.4.3 Expected Output**.

Returnthe integer value indicating where the data has to be indexed in the hash Table.  
**3.2.4.4 Success Criteria**   
successfully find the index value.

**3.2.6 Function: ProbeDec\_1 (char \*key)**

**3.2.6.1 Purpose and Procedure**

The purpose of this function is to decrement the index by linear probing if the collision is found from double hashing.

**3.2.6.2 Inputs**

Key-> The character Key value.

**3.2.6.3 Expected Output**

Returns the integer value ‘1’ indicating the value to which the Hash index decrement when collision is found.  
**3.2.6.4 Success Criteria**

Returns thePre-Decrement value correctly for double hashing.

**3.2.7 Function: ProbeDec\_2 (char \*key)**

**3.2.7.1 Purpose and Procedure**

The purpose of this function is to decrement the index if the collision is found from double hashing.

**3.2.7.2 Inputs**

Key-> The character Key value.

**3.2.7.3 Expected Output**

Returns the integer value indicating the value to which the Hash index decrement when collision is found.  
**3.2.7.4 Success Criteria**

Returns thePre-Decrement value correctly for double hashing.

**3.2.8 Function: ProbeDec\_3 (char \*key)**

**3.2.8.1 Purpose and Procedure**

The purpose of this function is to decrement the index if the collision is found from double hashing.

**3.2.8.2 Inputs**

Key-> The character Key value.

**3.2.8.3 Expected Output**

Returns the integer value indicating the value to which the Hash index decrement when collision is found.  
**3.2.8.4 Success Criteria**

Returns thePre-Decrement value correctly for double hashing

**Appendices**

**Test Forms:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Tested** | **Inputs** | **Expected output** | **Actual output** | **Pass/Fail** |
| **InitTable (HashStruct hashT[], int TableSize)** | **InitTable (T, 100)** | Initializes all struct array Data element value to zero. | Initializes all struct array Data element value to zero |  |
| **int HashInsert (HashStruct T[], char \*key, int data, int hNum, int dhNum);** | **int HashInsert (T, OBEC,** **36, 1, 1);** | Returns the collision count value for hashing1 and double hashing 1 function. | Returns the collision count value for hashing1 and double hashing 1 function |  |
| **int HashInsert (HashStruct T[], char \*key, int data, int hNum, int dhNum);** | **int HashInsert (T, OBIL,** **38, 2, 2);** | Returns the collision count value for hashing2 and double hashing 2 function. | Returns the collision count value for hashing2 and double hashing 2 function. |  |
| **Hash\_1(key)** | **Hash\_1(OBIL)** | Integer value  (O+B+I+L) % 100 | Integer value  (O+B+I+L) % 100 |  |
| **Hash\_2(key)** | **Hash\_2(key)** | (B+E)2 % 100 | (B+E)2 % 100 |  |
| **Hash\_3(key)** | **Hash\_3(OBON)** | (O\*B)% 100 | (B+E)2 % 100 |  |
| **ProbeDec\_2 (char \*key)** | **ProbeDec\_2 (OBON)** | Pre-Decrement  O+B+O+N/ 100 | Pre-Decrement  O+B+O+N/ 100 |  |
| **ProbeDec\_3 (char \*key)** | **ProbeDec\_2 (OBON)** | Pre-Decrement  (O\*B)+(O\*N)/ 100 | Pre-Decrement  (O\*B)+(O\*N)/ 100 |  |
| **ProbeDec\_1 (char \*key)** | **ProbeDec\_1 (OBON)** | Pre-Decrement  Value is 1 | Pre-Decrement  Value is 1 |  |