**Software Design Document**

Prepared By

Poornima Byre Gowda

[pg0018@uah.edu](mailto:pg0018@uah.edu)

Prepared for

Data Structures in C++

Computer Science Department

University of Alabama in Huntsville

**Table Of Contents**

1. System overview............................................................................................................ 3
2. Referenced Documents …………………………………………………………………………………………… 3
3. Architectural Design……………………………………………………………………………………………….. 3

3.1 Concepts of execution ………………………………………………………..…………………. 3

3.2 Code outline………………………………………………………………………………………….. 4

4.0 Detailed Design

4.1 Source file: ……………………………………………………………………………………………. 5

**Software Design Document**

**Programming Assignment 4**

**1.0 System Overview**  
  
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**

* Programming Assignment 1 Statement of Work.
* Dale, Nell and Teague, David, *C++ Plus Data Structures* 2nd ed. 2001.
* Dr. Coleman’s CS 221 Lecture Slides.

**3.0 Architectural Design**

**3.1 Concept of Execution**

This programming assignment you will have the opportunity to study the effects of different hash functions on the efficiency of inserting data into a hash table. The student shall define, develop, document, prototype, test, and modify as required the software system.  
  
**3.2 Abstract Data Type**

This program will be implemented using the abstract datatype which a Structure named as “HashStruct”. The purpose of using this structure HashStruct here is to create the character datatype ‘Key’, integer variable ‘Data’.

**3.3 Code Outline**

The classes and headers used in this program are listed below:

**Class: Main**

**Files: Hashing.h , main.cpp.**

These files will create the instances of the struct “HashStruct” to store the data with respective to the key value and store it in Hash Table base on the Hash Index value. Hash index is calculated based the key value with respective calculation methods.

**Functions:**

void InitTable(HashStruct hashT[], int TableSize)

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

int Hash\_1(char \*key)

int Hash\_2(char \*key)

int Hash\_3(char \*key)

int ProbeDec\_1(char \*key)

int ProbeDec\_2(char \*key)

int ProbeDec\_3(char \*key)

**4.0 Detailed Design**

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

**4.1.2 Function: void InitTable(HashStruct hashT[], int TableSize)**

**4.1.2.1 Purpose**

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

**4.1.2.2 Arguments**  
 HashStruct**\_**T **(** variableof type HashStruct)

TableSize **(**integer variable indicating size of the Table)

**4.1.2.3 Return value**

Function returns no value.  
**4.1.2.4 Function outline in pseudocode**strcpy(hashT[i].key, EMPTYKEY)

hashT[i].data = 0

**4.1.2.5 Traceability**

It partially fulfils the requirement of 2.1.2.

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

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

**4.1.3.2 Arguments**

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.

**4.1.3.3 Return value**

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

**4.1.3.4 Function outline in pseudocode**Declare integer testNum and set to (hNum \* 3) + dhNum

Initialize integer colCount to 0

Declare integer hashIndex, probeDec

switch(testNum)

Case 1. Calls function Hash\_01 to find Hash index.

Calls function ProbeDec\_1 for probe increment

Case 2. Calls function Hash\_01 to find Hash index.

Calls function ProbeDec\_2 for probe increment

Case 3. Calls function Hash\_01 to find Hash index.

Calls function ProbeDec\_3 for probe increment

Case 4. Calls function Hash\_02 to find Hash index.

Calls function ProbeDec\_1 for probe increment

Case 5. Calls function Hash\_02 to find Hash index.

Calls function ProbeDec\_2 for probe increment

Case 6. Calls function Hash\_02 to find Hash index.

Calls function ProbeDec\_3 for probe increment

Case 7. Calls function Hash\_01 to find Hash index.

Calls function ProbeDec\_1 for probe increment

Case 8. Calls function Hash\_01 to find Hash index.

Calls function ProbeDec\_2 for probe increment

Case 9. Calls function Hash\_01 to find Hash index.

Calls function ProbeDec\_3 for probe increment

Insert key and data into table

Return the total number of collisions and the diagram of the table

**4.1.3.5 Traceability**

It partially fulfils the requirement of 2.1.2.

**4.1.4 Function: int Hash\_1(char \*key)**

**4.1.4.1 Purpose**  
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).

**4.1.4.2 Arguments**

Key-> The character Key value

**4.1.4.3 Return value**

Return the integer value. i.e. Hash\_index  
**4.1.4.4 Function outline in pseudocode**

Set integer index to (key[0] + key[1] + key[2] + key[3] ) % TABLESIZE

Return index

**4.1.4.5 Traceability**

It partially fulfils the requirement of 2.1.2.

**4.1.5 Function: int Hash\_2(char \*key)**

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

**4.1.5.2 Arguments**

Key-> The character Key value

**4.1.5.3 Return value**

Return the integer value. i.e. Hash\_index  
**4.1.5.4 Function outline in pseudocode**

Set integer index to (key[1] + key[2])2 % TABLESIZE

Return index

**4.1.5.5 Traceability**

It partially fulfils the requirement of 2.1.2.

**4.1.6 Function: int Hash\_3(char \*key)**

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

**4.1.6.2 Arguments**

Key-> The character Key value.

**4.1.6.3 Return value**

Return the integer value. i.e. Hash\_index  
**4.1.6.4 Function outline in pseudocode**

Set integer index to (key[0] \* key[1]) % TABLESIZE

Return index

**4.1.6.5 Traceability**

It partially fulfils the requirement of 2.1.2.

**4.1.7 Function: int ProbeDec\_1(char \*key)**

**4.1.7.1 Purpose**  
Calculate the increment value based on linear probing.

**4.1.7.2 Arguments**  
Key-> The character Key value.

**4.1.7.3 Return value**

There is an integer return value i.e. Pre-Decrement.  
**4.1.7.4 Function outline in pseudocode**

Return 1.

**4.1.7.5 Traceability**

It partially fulfils the requirement of 2.1.2.

**4.1.8 Function: int ProbeDec\_2(char \*key)**

**4.1.8.1 Purpose**  
Calculate the increment value based on average division method technique.

**4.1.8.2 Arguments**

Key-> The character Key value.  
**4.1.8.3 Return value**

There is an integer return value i.e. increment.  
**4.1.8.4 Function outline in pseudocode**

ProbeDec= (key[0]+ key[1]+ key[2]+ key[3] )/ TABLESIZE.

ProbeDec= max(1, q)

**4.1.8.5 Traceability**

It partially fulfils the requirement of 2.1.2.

**4.1.9 Function: int ProbeDec\_3(char \*key)**

**4.1.8.1 Purpose**  
Calculate the increment value based on average division method technique.

**4.1.8.2 Arguments**

Key-> The character Key value.  
**4.1.8.3 Return value**

There is an integer return value i.e. increment.  
**4.1.8.4 Function outline in pseudocode**

ProbeDec= ((key[0]\* key[1] )+ (key[2]\* key[3] )) / TABLESIZE.

ProbeDec= max(1, q)

**4.1.8.5 Traceability**

It partially fulfils the requirement of 2.1.2.