School of Informatics

DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTING

3SFE504: Object oriented programming

COURSEWORK

Student Id	*Student First Name*	*Student Surname*
2007053	**Kavindya**	**Isuru**
2007020	**Malshani**	**Nanayakkara**

Module Leader: **Mrs.Udayangi Perera**

CONTENTS

INTRODUCTION

This coursework was done in teams' of two group members.

Group V was made of up of:
N.T. I Kavindya and Malshani Nanayakkara

In this project, we were asked to create a dictionary object, which was capable of storing words and its definitions. Each word could contain more than one definition. The dictionary object we created should contain the facilities of viewing the words in the dictionary, adding new words, searching new words and printing it to a file.

The words and their definitions would be stored in an external file called 'defs.dat'. So every time, our application is run, the words and their definitions should be read in. We each created a hash table to accommodate these. Student A's part, done by Kavindya, reads in the word and stores them using the linear rehashing technique. Student B's part, done by Malshani, stores the words in the hash table using the independent rehashing technique.

IMPLEMENTATION

GROUP WORK

Definition and List Class files

Definition.h

```
//Definition.h
#ifndef Definition_h
#define Definition_h

#include <string>
#include <cstdlib>

using namespace std;

class Definition
{
	public:
		Definition();		//constructor (initialisation)
		~Definition();

		//setter methods
		void setDefinition(string def);
		void setNext(Definition*);

		//getter methods
		string getDefinition();
		Definition* getNext();

	private:
		string definition;
		Definition* next;


};

#endif
```

Definition.cc

```
#include "Definition.h"
#include <string>
#include <cstdlib>
#include <iostream>

using namespace std;
```

```
Definition::Definition()
{
	definition = "NULL";
	next = NULL;
}

Definition::~Definition()
{
}

void Definition::setDefinition(string def)
{
	definition=def;
}

void Definition::setNext(Definition* n)
{
	next=n;
}

string Definition::getDefinition()
{
	return definition;
}

Definition* Definition::getNext()
{
	return next;
}
```

List.h

```
//List.h

#ifndef List_h
#define List_h

#include <string>
#include "Definition.h"

using namespace std;

class List
{
	public:
		List();
```

```
        ~List();

        //inserts definition at the front of the list
        void insertH(string definition);

        //whether list is empty
        bool isEmpty() const;

        //display the list contents
        void printList() const;


    private:
        Definition* head;       //points to head of list
};

#endif
```

List.cc

```
//List.cc

#include "Definition.h"
#include "List.h"
#include <iostream>

using namespace std;

//constructor - initialises empty list
List::List()
{
    head = NULL;
}

List::~List()
{
    Definition* current;
  Definition* i;

  i = head;
  while (i != NULL)
  {
    current = i;
    i = i->getNext();
    delete current;
  }
}
```

```
bool List::isEmpty() const
{
	return head==NULL;
}

void List::insertH(string definition)    //inserts node at head
{
	Definition* n = new Definition;

	n->setDefinition(definition);
	n->setNext(head);
	head = n;
}


void List::printList() const
{
	if ( isEmpty() )
	{
		cout << "The List is empty" << endl;
	}
	else
	{
		Definition* n = NULL;

		for ( n=head; n!=NULL; n = n->getNext() )
		{
			cout << "\t" << n->getDefinition() << endl;		//display data
value
		}
		cout << endl;
	}
}
```

STUDENT A -
LINEAR REHASHING TECHNIQUE

Student A

//Definition .h (header file)

```
//Definition.h
#include<string>
using namespace std;

class Definition
{
public:
	Definition();
	~Definition();
	void setDefinition(string); //setter method
	void setNext(Definition*); //setter method
	string getDefinition(); //getter method
	Definition* getNext(); // Definition pointed to getNext method

private:
	string definition;
	Definition* next;
};
```

//Definition.cpp

```
#include<iostream>
#include "Definition.h"
#include<string>

using namespace std;

Definition::Definition()
{
	definition="";
	next=NULL;
}
Definition ::~Definition()
{
}
void Definition::setDefinition(string d)
{
	definition = d;
}
void Definition::setNext(Definition* n)
{
	next = n;
}
string Definition::getDefinition()
{
	return definition;
}
Definition* Definition::getNext()
{
	return next;
}
```

//List.h

```
#include <string>
#include "Definition.h"

using namespace std;

class List
{
public:
        List();
        ~List();

        void insertAtHead(const string);
        int  getHead() const;
        void printList() const;
        bool isln (const int) const;
        bool isEmpty() const;
private:
        Definition* head;
};
```

//List.cpp

```
#include<iostream>
#include "List.h"
#include <string>
using namespace std;

List::List()
{
	head = NULL;
}
List::~List()
{
}
bool List::isEmpty()const
{
	return head == NULL;
}
void List::insertAtHead(const string value)
{
	Definition* p = new Definition;
	p->setDefinition(value);
	p->setNext(head);
	head=p;
}
void List::printList() const
{
	if(isEmpty())
	{
		cout<<"list empty"<<endl;
	}
	else
	{
		Definition* p =NULL;
		for(p=head ; p != NULL ; p=p->getNext())
		{
			cout<<p->getDefinition()<<endl;
		}
	}
}
```

//LinHTable.h

```
#include <string>
#include "List.h"

using namespace std;
const int TABLE_SIZE=29;

class LinHTable
{
public:
	LinHTable();
	~LinHTable();
	void put(string word, string def);

	List get(string word);
	void traverse();//Traverse until non-empty dic entry
	void search(string); // search t he word
	//void remove();
private:
	struct b
	{
		string word;
		List defList;
	}dictionary[TABLE_SIZE];
};
```

//LinHTable.cpp

```
#include<iostream>
#include<string>
#include "LinHTable.h"

using namespace std;

LinHTable::LinHTable()
{
}
LinHTable::~LinHTable()
{
}
// calculating the hash value for the string word
int HashVal(string str)
{
        int n=0;
        char alp[]={'a','b','c','d','e','f','g','h','i','j','k','l','m'
                                ,'n','o','p','q','r','s','t','u','v','w','x','y','z'};

        for(int i=0; i<=str.length(); i++)
        {
                for(int ch=0; ch <=25 ; ch++) // calculate string ch between A-Z
                {
                        if(str[i]==alp[ch])
                        {
                                n+=ch+1;
                        }
                }
        }
return (n%TABLE_SIZE); //calculate remain table size
}
//puting the word and definition to the hash table
void LinHTable::put(string word, string def)
{
int dic=HashVal(word);
        if(dictionary[dic].word=="" || dictionary[dic].word==word)
        {
                        dictionary[dic].word=word;
                dictionary[dic].defList.insertAtHead(def);
        }
        else if(dictionary[dic].word!="")
        {
   int main=dic; //initialise head

        int t=0;
```

```
                do
                {
                        dic++;
                        if(dic>=TABLE_SIZE)
                        {
                                dic=dic-TABLE_SIZE;
                        }
                        if(dictionary[dic].word==word)
                        {
                                dictionary[dic].word=word;
                                dictionary[dic].defList.insertAtHead(def);
                                t++;
                        }
                }while(dictionary[dic].word!="" && main!=dic);
                        if(t==0)
                        {
                                dictionary[dic].word=word;
                                dictionary[dic].defList.insertAtHead(def);
                        }
        }
}
List LinHTable::get(string w)
{
        int key=HashVal(w);

        while(dictionary[key].word!=w)
        {
                key++;// key  is represents increment to be used .
        }
        return dictionary[key].defList;
}
void LinHTable::traverse()
{
   //displaying word and defini
        for(int h=0 ;h<TABLE_SIZE; h++)
        {
                if(dictionary[h].word!="")
                {
                        cout<<dictionary[h].word<<": "<<endl;
                        get(dictionary[h].word).printList();
                        cout<<endl;
                }
        }
}
// to search , it will display the search word
void LinHTable::search(string str)
{
        int inc=0;
        for(int key=0 ; key<TABLE_SIZE; key++)
```

```
        {//hash function to calculate the table index for search key.
                if(dictionary[key].word==str)
                {
                        get(str).printList();
                        inc++; // inc is represents increment to be used .
                }
        }
        if (inc==0)
                cerr<<"Word Not found!"<<endl;
}

//remove word from table
//void LinHTable::remove(string word)
//{
//      int location = hashVal(word);
//      dictionary[location].word = "";
//      dictionary[location].defList.clear();

//}
```

//Main.cpp

```
#include<iostream>
#include <string>
#include<fstream>
#include "LinHTable.h"
using namespace std;

struct Def
{
	char definition[1000];// get definition from the file defs.dat
	char inDefin[1000];//getting users inputs
}HaFile[TABLE_SIZE];

string wrd[TABLE_SIZE];	//get the word from  defs.dat file
string Inwrd;		// getting user input word
int Caline=0;	//count number of words in the dictionary
// sort the word in alphabetical order
void bubbleSort(string arr[],int size)
{
	int i, j;
	string temp;
	for(i=size-1;i>0;i--)
	{
		for(j=0;j<i;j++)
		{
			if(arr[j]>arr[j+1])
			{
				temp=arr[j];
				arr[j]=arr[j+1];
				arr[j+1]=temp;
			}	//end if
		}	//end for
	}	//end for
}	//end BubbleSort

void option(){ //options for seletion
		cout<<"\n\n\t $$$####**** WELCOME TO DICTIONARY
****####$$$\n"<< endl;
		cout<<"\t     .*********************************. "<<endl;
		cout<<"\t    # *                              * #"<<endl;
		cout<<"\t    # *    1. Display The Directory   * #"<<endl;
		cout<<"\t    # *    2. Search your word        * #"<<endl;
		cout<<"\t    # *    3. Sorted your dictionary  * #"<<endl;
		cout<<"\t    # *    4. Inserting new word      * #"<<endl;
		cout<<"\t    # *    5. Close the Dictionary    * #"<<endl;
		cout<<"\t    # *                              * #"<<endl;
		cout<<"\t      *********************************"<<endl;
```

```
}
//convert to lowercase
void toLower(basic_string<char>& s)
{
  for (basic_string<char>::iterator p = s.begin( );
  p != s.end( ); ++p)
                {
                        *p = tolower(*p);
                }
}

//check the word
bool charchk(string str) //check the word is valid or not .
{
        int cha=0;

        char alp[]={'a','b','c','d','e','f','g','h','i','j','k',
                                'l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};// makes an array
for 26 elements to insert alphabetics
        for(int c=0 ; c<=str.length() ; c++)
        {
                for(int Def=0 ; Def<=25 ; Def++)
                if(str[c]==alp[Def])
                {
                        cha++;
                }

        }
        if(cha==str.length())
                return true;
        else
                return false;
}

void readDicf(LinHTable& b)
{
  //geting the lines of file
        ifstream data("defs.dat");

        if(!data.is_open())
        {
                cerr<<"Unable to open the file!!"<<endl;
                exit (1);
        }
                //Read the data from defs.dat file
                while(!data.eof())
                {
                        for(int rd=0 ; rd<TABLE_SIZE ; rd++)
```

```
                        {
                                data>>wrd[rd];
                                data.getline(HaFile[rd].definition,1000,'\n');
                                if(charchk(wrd[rd]))
                                b.put(wrd[rd],HaFile[rd].definition);
                                if(wrd[rd]!="")
                                Caline++;
                        //reading & saving the string to list of the file
                        }
                }
        data.close();
}
//Display the dictionary
void displayAll(LinHTable& da)
{
        da.traverse();
}
// search the word from the hash table
void Searchwd(LinHTable& da)
{
        cout<<"Enter the word you  want to search : ";
        cin>>Inwrd;
        toLower(Inwrd);         //convert to lowercase
        da.search(Inwrd);       //check the word
}
bool twd(string w1, string w2)//check the word to word .
{
        if(w1==w2)
        {
                return false; // if word to word equal then return false
        }
        else // if not return true
                return true;
}
//sorting the dictionary
void sortString(LinHTable& Ss)
{
        bubbleSort(wrd,Caline);

        for(int rd=0 ; rd<Caline; rd++)
        {
                if(charchk(wrd[rd]))
                {
                        if(twd(wrd[rd],wrd[rd+1]))
                        {
                                cout<<wrd[rd]<<endl;
                                Ss.get(wrd[rd]).printList();
                                cout<<endl;
                        }
```

```
                }
        }
}
//insert new word & definition into the list
void insert()
{
        cout<<"Enter the word you want to add : ";
        cin>> Inwrd;
        cin.ignore();
        cout<<"Enter the Definition for the word(end with [.]) : ";
        cin.getline(HaFile[0].inDefin, 1000, ' . ');

        toLower(Inwrd); // converting user input to lowercase
        ofstream wri("defs.dat");

        for(int d=0; d<Caline; d++)
        {
                wri<<wrd[d];
                wri<<HaFile[d].definition<<endl;
        }
        wri<<Inwrd;
        wri<<" "<<HaFile[0].inDefin<<"."<<endl;
        cout<<"You inserted word successfully"<<endl<<endl;
}
int main()
{
        LinHTable ht;
        readDicf(ht);
        char menu;
        option(); //call the option method
        do{
                cout<<endl<<"Enter your choice[1-5]: ";
                cin>>menu;
                cout<<endl<<endl;
                if(!( menu=='1' ||menu=='2' ||menu=='3'||menu=='4'||menu=='5'))
                {
                        cerr<<"Invalied choice, Please enter between[1-5]"<<endl;
                }
                if(menu=='1')
                {
                        displayAll(ht);
                        option(); //call the option method
                }
                else if(menu=='2')
                {
                        Searchwd(ht);
                        option(); //call the option method
                }
                else if(menu== '3')
```

```
		{
			sortString(ht);
			option(); //call the option method
		}
		else if(menu=='4')
		{
			insert();
			option(); //call the option method
		}
	}while(menu!='5');
	cout<<"\t\t\t------Thank you ----\n";
	cout<<"\t\t---------------*******----------------\n";
	return 0;
}
```

Assumptions

void Option()
I create this function for options. There will display 5 options for user needs

void LinHTable::traverse()
here display the perfect dictionary there words are ordered by alphabetical order I call sorting function there for this purpose . There I used Bubble sort for this sorted function.

void sortString(LinHTable& Ss)
here the words are sorted as alphabetical order & I used bubble Sort method for sort the strings.Here the bubble sort check elements in array and get the first letter from the word and sorted as alphabetical order.

void insert()
Here insert word & definition inside the hash table before insertion it will check the empty table & put word and definition there & its check the word, That if its already there,then again insert word & definition different places. But when we are searching in the run program, there show only one word and many definitions.

List LinHTable::get(string w)
here the list of definition associated with key word return null list if word is not present in the table.So list is return.

int HashVal(string str)

Here I declare an array of alphabets (a-z) for hash function. The hash function will get the string, and in the for loop it will get the value or each letter and add into "variable n", After getting the total of "variable n" I divide that by table size and taking the reminder, then the num will not exist the table limit 29.

References

Recipe 4.12. Converting a String to Lower- or Uppercase
example 4-20. Converting a string's case
C++ Cookbook
By Jeff Cogswell, Christopher Diggins, Ryan Stephens, Jonathan Turkanis
Publisher: O'Reilly
Pub Date: November 2005
ISBN: 0-596-00761-2

STUDENT B-
INDEPENDENT REHASHING TECHNIQUE

CODING

As a group, we created the Definition.cc and List.cc files together.

Definition.h

```
//Definition.h
#ifndef Definition_h
#define Definition_h

#include <string>
#include <cstdlib>

using namespace std;

class Definition
{
	public:
		Definition();		//constructor (initialisation)
		~Definition();

		//setter methods
		void setDefinition(string def);
		void setNext(Definition*);

		//getter methods
		string getDefinition();
		Definition* getNext();

	private:
		string definition;
		Definition* next;


};

#endif
```

Definition.cc

```
#include "Definition.h"
#include <string>
#include <cstdlib>
#include <iostream>

using namespace std;

Definition::Definition()
{
	definition = "NULL";
	next = NULL;
}

Definition::~Definition()
{
}

void Definition::setDefinition(string def)
{
	definition=def;
}

void Definition::setNext(Definition* n)
{
	next=n;
}

string Definition::getDefinition()
{
	return definition;
}

Definition* Definition::getNext()
{
	return next;
}
```

For my part, I have edited the List class to accommodate a few functions, that I require in my part. They have been included below:

List.h

```
//List.h
// Student: Malshani Nanayakkara
// Stu_No: 2007020

#ifndef List_h
#define List_h

#include <string>
#include "Definition.h"

using namespace std;

class List
{
	public:
		List();		//constructor - initialise empty list
		~List();	//destructor

		Definition* getHead();
		string getList();

		//inserts definition at the front of the list
		void insertH(string definition);
		void insertR(string definition);

		//whether list is empty
		bool isEmpty() const;

		//display the list contents
		void printList() const;
		void printListToFile() const;


	private:
		Definition* head;	//points to head of list
};

#endif
```

List.cc

```
//List.cc
// Student: Malshani Nanayakkara
// Stu_NO: 2007020

#include "Definition.h"
#include "List.h"
#include <stddef.h>
#include <iostream>
#include <fstream>

using namespace std;

//constructor - initialises empty list
List::List()
{
        head = NULL;
}

List::~List()
{
        Definition* current;
   Definition* i;

   i = head;
   while (i != NULL)
   {
      current = i;
      i = i->getNext();
      delete current;
   }
}

Definition* List::getHead()
{
        return head;
}

string List::getList()
{
        string output, z;
        string blank="\n";

        Definition* n = NULL;

                for ( n=head; n!=NULL; n = n->getNext() )
                {
                        output += blank + n->getDefinition();                   //display data
value
```

```
                        //return  string(n->getDefinition());
                        //cout << n->getDefinition() << endl;
                }

        //free(n);

        return output;
}

bool List::isEmpty() const
{
        return head==NULL;
}

void List::insertH(string definition)   //inserts node at head
{
        Definition* n = new Definition;

        n->setDefinition(definition);
        n->setNext(head);
        head = n;
}

void List::insertR(string definition)   //inserts node at rear
{
        Definition* n = new Definition;         //new node

        n->setDefinition(definition);
        n->setNext(NULL);

        Definition* current;    //node to traverse the list
        current = head;                 //start at beginning of list

        while(current->getNext()!=NULL)
        {
                current = current->getNext();
        }

        current->setNext(n);   //sets last node to point to the new node;
}

void List::printList() const
{
        if ( isEmpty() )
        {
                cout << "The List is empty" << endl;
        }
        else
```

```
	{
		Definition* n = NULL;

		for ( n=head; n!=NULL; n = n->getNext() )
		{
			cout << "\t" << n->getDefinition() << endl;		//display data
value
		}
		cout << endl;
	}
}

void List::printListToFile() const
{
	ofstream writefile("dict.dat", ios::app);

	if(!(isEmpty() ))
	{
		Definition* n = NULL;

		for ( n=head; n!=NULL; n = n->getNext() ) {
			//display data value
			writefile <<"\t"<< n->getDefinition() << endl;
		}

	}

	writefile.close();
}
```

As I have changed the List.cc file, my test class has also changed so that I could test this new function.

List_test.cc

```
//List_test.cc
// Student: Malshani Nanayakkara
// Stu_NO: 2007020

#include <iostream>
#include <stddef.h>
#include <fstream>
#include "List.h"

using namespace std;

int main()
{
	List l;
	l.insertH("hello");
	l.insertH("world");
	l.insertH("good bye");
	l.insertR("tata");
	cout <<"\nCalling printlist() function: " << endl;
	l.printList();

	cout <<"\nCalling getlist() function:" << endl;
	cout << "1" << l.getList() <<endl<<endl;

	ofstream write("test.txt");
	write << "List items:\n" << endl;
	write << l.getList() << endl;
	write << "End of File! "<< endl;
	write.close();


	List l2; //empty list
	l2.printList();
}
```

Included below is my IndepHTable files:

IndepHTable.h

```
//IndepHTable.h
// Student: Malshani Nanayakkara
// Stu_NO: 2007020

#ifndef IndepHTable_h
#define IndepHTable_h
#include <string>
#include "List.h"

const int TABLE_SIZE = 29;

using namespace std;

class IndepHTable
{
	public:
		IndepHTable();			//constructor - create empty table
		~IndepHTable();			//destructor

			//enter data into table
		void load_dict();
		void put (string word, string def);
		void add(string, string);
		void sort_arr(int);

			//return the list of definition(s) associated with 'word'
		int getIndex(string);
		List get(string word);
		//void get(string word);

			//display non-empty hash entries
		void print();

		bool valid(string);
		int hash(string, char);

	private:
		struct
		{
			string word;
			List defList;
		}dictionary[TABLE_SIZE];
};

#endif
```

IndepHTable.cc

```
//IndepHTable.cc
// Student: Malshani Nanayakkara
// Stu_NO: 2007020

#include "IndepHTable.h"
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <cstdlib>

using namespace std;

//constructor
IndepHTable::IndepHTable()
{
	dictionary->word="";
}

IndepHTable::~IndepHTable()
{
}

//to load dictionary
void IndepHTable::load_dict()
{
	ifstream readfile ("dictionary.txt");
	string word, line;

	if (readfile.is_open())
	{
		while (!readfile.eof() )
		{
			readfile >> word;
			getline (readfile,line);
			put(word, line);

		}
		readfile.close();
	}
	else
		cout << "Unable to open dictionary file"<<endl;
}


bool IndepHTable::valid(string word)
```

```
{
	char word_arr[20];		//assuming longest word is less than or equal to 20 characters
	strcpy(word_arr, word.c_str() );	//converts string to char array

	int sz = strlen(word_arr);	//gets the number of characters in the array
	int flag = 0;

	for (int i=0; i<sz; i++) {

		if (! ((word_arr[i]>=65 && word_arr[i]<=90) || (word_arr[i]>=97 && word_arr[i]<=122)) ) //checks if character is either an upper case or lower case letter
			flag=1;		//an invalid character has been entered}
	}

	if (flag==0)
			return true;
	else
		return false;
}

		//returns the index for the hash table
int IndepHTable::hash(string word, char option)
{
	char word_arr[20];	//assuming longest word is less than or equal to 20 characters
	int hash_index=0, rehash_inc=0, total=0;

	strcpy(word_arr, word.c_str() );		//converts string to a char array

	int sz = strlen(word_arr);		//gets the number of characters in the array.

	for (int i=0; i<sz; i++) {
		total += (int)word_arr[i];		//gets the ASCII equivalent and sums up all the characters' values
	}

		if (option=='h')
		{
			hash_index = total % TABLE_SIZE; //calculates hash index using the division-remainder technique
			//cout << word << " " << "index: " << hash_index << endl;
			return hash_index;
		}
		else if (option=='r')
		{
			rehash_inc = total % 23;		//calculates the increment for rehashing by division-remainder technique with the next smallest prime number
```

```
                        if (rehash_inc==0)              //since increment value can not be
value, it is made to one.
                                rehash_inc = 1;

                        return rehash_inc;
                }

}

void IndepHTable::put(string word, string def)          //adds word to hash table
{
        if(valid(word))         //checks whether word contains any non-alphabetic
characters
        {
                int index = hash(word, 'h');
                int increment = hash(word, 'r');

                int flag = 0;   //flag to indicate whether word has been added to hash table
or not. flag=0, means its not added.

                do{
                        if ( dictionary[index].word=="" ||
word.compare(dictionary[index].word)==0 ) {         //if the slot is empty  OR if the the
same word is already stored at that index
                                dictionary[index].word = word;
                                dictionary[index].defList.insertH(def);
                                flag = 1;
                        }
                        else {                          //different word is stored in the index
so index is changed by adding increment.
                                index += increment;

                                if (index>=TABLE_SIZE)              //checks if index is
out of table range
                                        index -= TABLE_SIZE;
                        }

                } while (flag==0);
        }
        else
                cout << " Error: The word " << word << " is invalid! Can not be added to
dictionary" << endl;

}

void IndepHTable::add(string new_word, string new_def)
{
        if(valid(new_word)) {
                put(new_word, new_def);
```

```
                ofstream writefile ("dictionary.txt",ios::app);
                writefile << "\n" << new_word << " " << new_def;
                writefile.close();
        }
        else
                cout << "The word " << new_word << "is invalid! " << endl;


}

void IndepHTable::sort_arr(int option)
{
        string arr[TABLE_SIZE];
        int sz =TABLE_SIZE;

        List result;
        int x;

        for (int i=0; i<TABLE_SIZE; i++)    {
                arr[i] = dictionary[i].word;
        }

        sort(arr, arr+sz);

        if (option==0)
        {
                //writes to file dict.dat
                ofstream writefile ("dict.dat");
                for (int j=0; j < TABLE_SIZE; j++)
                {
                        if (arr[j]!="")
                        {
                                //get index of where the word is stored in the hash table
                                x = getIndex(arr[j]);
                                writefile << "\n" << arr[j] << " : "<<
dictionary[x].defList.getList()<<"\n";
                        }
                }
                writefile.close();

                cout << "\nThe Dictionary has been written to dict.dat file..." <<endl;
        }
        else
        {
                for (int k=0; k < TABLE_SIZE; k++)
                {
                        if (arr[k]!="")
                        {
```

```
                                //get index of where the word is stored in the hash table
                                x = getIndex(arr[k]);
                                cout<< "\n" << arr[k] << " : "<<
dictionary[x].defList.getList()<<"\n";
                        }
                }

        }
}

int IndepHTable::getIndex(string findword)
{
        //linear search in hash table
        int index;
        for (int i=0; i< TABLE_SIZE; i++)
        {
                if(findword.compare(dictionary[i].word)==0)
                { index = i; }
        }

        return index;   //as words are taken from the dictionary and then sorted, an index
will always be returned.
}

List IndepHTable::get(string searchword)
{
        //search hash table for 'word' and return list of definitions
        List result;

        if (valid(searchword))
        {
                int index = hash(searchword, 'h');
                int increment = hash(searchword, 'r');
                int count = 0;

                while ( (searchword.compare(dictionary[index].word)!=0 )  &&
(count<TABLE_SIZE) )        //exits loop if word is found
                {
                        index += increment;             //increments index to next location

                        if (index>=TABLE_SIZE)              //checks if index is out of
table range
                                index -= TABLE_SIZE;

                        count++;        //counts the number of records that have been
traversed. if it is equal to TABLE_SIZE, all records in hash table has been traversed!
                }

                if(searchword.compare(dictionary[index].word)==0 )
```

```
		{
			result = dictionary[index].defList;
			//result.printList();
		}
	}
	else
		cout << "The search word you have entered contains non-alphabetic
characters and is incorrect!" << endl;

	return result;
}

void IndepHTable::print()
{
	for (int i=0; i<TABLE_SIZE; i++)
	{
		if (dictionary[i].word!="") {
			cout << dictionary[i].word << " :\n";
			dictionary[i].defList.printList();	//prints out list
		}

	}
}
```

This file contains the menu and is equivalent of a main.cc file.

Indep_test.cc

```
//Indep_test.cc
// Student: Malshani Nanayakkara
// Stu_NO: 2007020

#include <iostream>
#include <fstream>
#include "IndepHTable.h"

using namespace std;

int main()
{
	IndepHTable Dict;
	char exit;

	Dict.load_dict();	//loads dictionary

	do{
	//MENU
	cout << endl<<endl;
	cout << "\t/-----------------------------------------------------\\"<<endl;
	cout << "\t|\t		DICTIONARY\t\t\t|"<<endl;
	cout << "\t\\-----------------------------------------------------/"<<endl<<endl;
	cout << "Menu options: "<< endl;
	cout << "\n1) Display Dictionary";
	cout <<" \n2) Search For A Word";
	cout << "\n3) Add A New Word To Dictionary";
	cout << "\n4) View Sorted Dictionary";
	cout << "\n5) Write Dictionary To File";
	cout << "\n6) Exit\n "<< endl;
	cout << "Please enter your selection: " ;

	int option;
	cin >> option;


	if (option==1)
	{
		cout << "\n\t---------------DISPLAY DICTIONARY---------------"<<endl;
			Dict.print();
	}
	else if (option==2)
	{
			//search and retrieves definitions for a given word
```

```
        cout << "\n\t---------------SEARCH FOR A WORD---------------"<<endl;
        string searchword;
        List defns;

        cout << "Enter search word : ";
        cin >> searchword;
        cout << "\nResults:" <<endl;
        cout << "\n" << searchword << " : ";
        Dict.get(searchword).printList();

}
else if (option==3)
{
        cout << "\n\t----------------ADD NEW WORDS----------------"<<endl;
        string new_w;
        string new_d;

        cout << "Enter a new word : " ;
        cin >> new_w;
        cin.get();          //get the 'enter' key press
        cout << " Enter its definition: " ;
        getline(cin, new_d);

        Dict.add(new_w, new_d);

        cout << "\nWord has been added! "<< endl;

}
else if(option==4)
{
        cout << "\n\t---------------SORTED DICTIONARY---------------"<<endl;
        Dict.sort_arr(1);
}
else if(option==5)
{
        Dict.sort_arr(0);
}
else
{
        if (option!=6)
        {
                cout << "Invalid menu option!"<< endl;
        }
}

cout << "\n Do you wish to exit the dictionary application?(y/n) :" ;
cin >> exit;

}while (exit=='n'|| exit=='N');
```

```
        cout << "\nExiting..." << endl;

        return 0;
}
```

TESTING

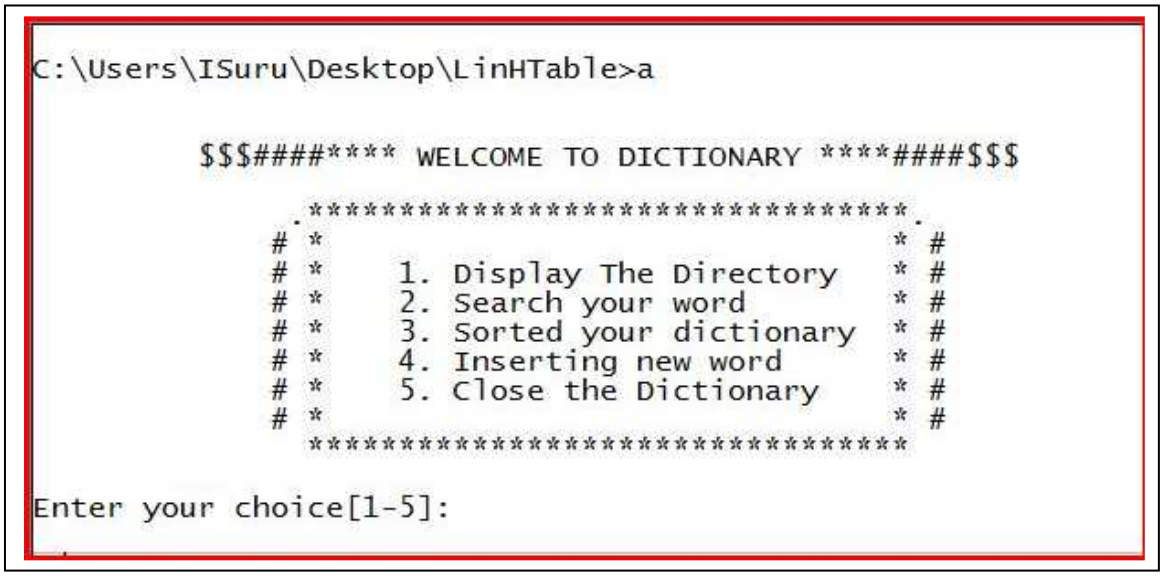
Testing part

Student A

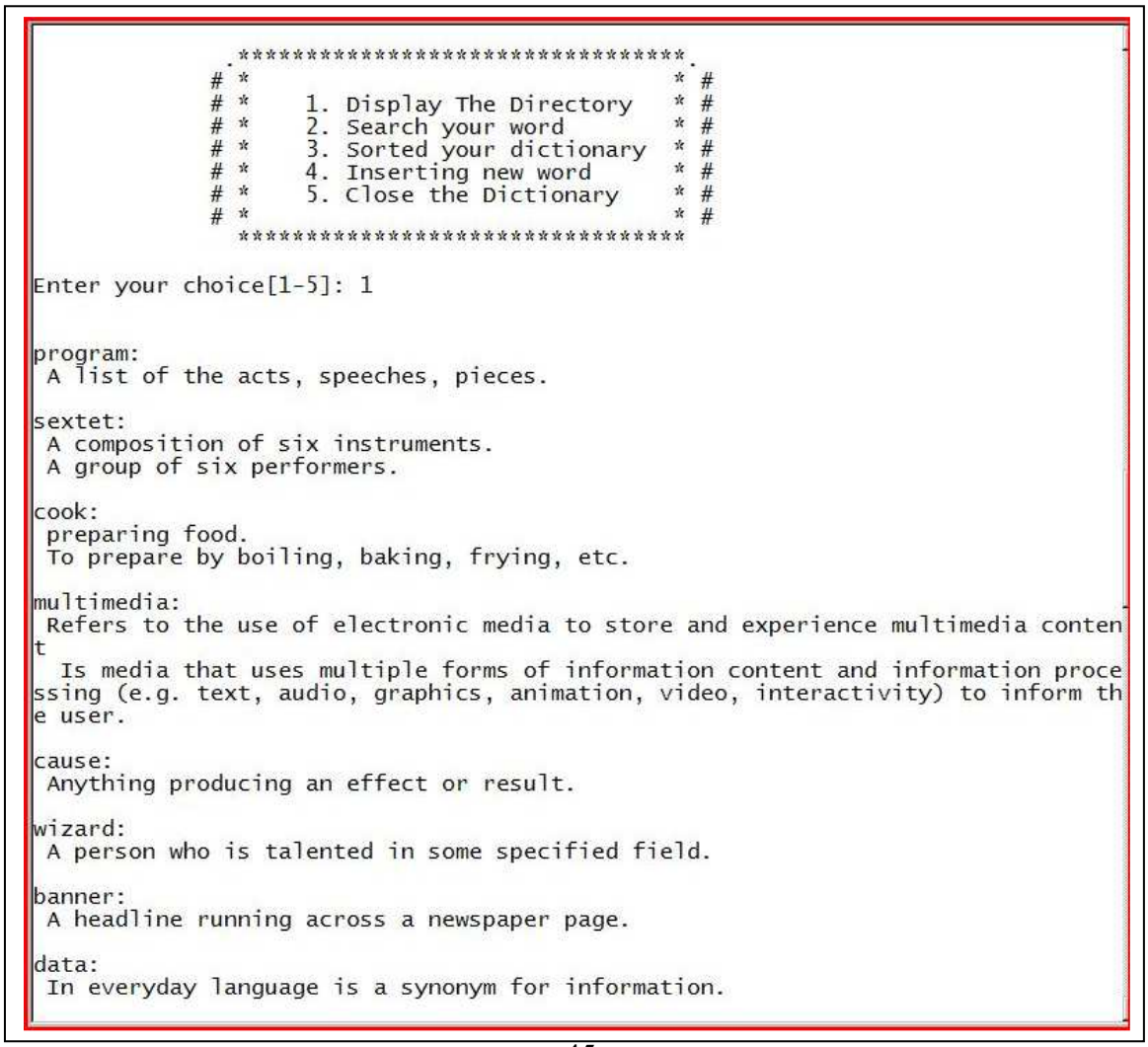
Option	**Expected out put**	**Actual output**	**Screen Shot No**	**Comments**
When select option 1	Display the dictionary	Display the dictionary	1	Ok
When select option 2	The program says the user to enter a word to be searched.	The program should ask for a word to insert and find.	2	Ok
When select option 3	Display the sorted words from the dictionary	The program should sorted words.	3	Ok
When select option 4	The program should ask to insert new words.	The program should ask to insert new words.	4	Ok
When select option 5	The program should be terminated	The program should be terminated	5	Ok
When the user selects an illegal option.	Then Display error massage as “Invalid choice, Please enter between[1-5]”	Should Display error massage	6	Ok
When the user enters a word which is not in the table.	Displays a error massage as “Word Not found!”	Display a error massage as “Word Not found!”	7	Ok
When the file has several definitions for the same word.	the definitions are different, the program enters them to the table.	The program should enter them to the table.	8	Ok
When the file “defs.dat” is not available.	Display the error massage “Unable to open the file!!” & should terminate the program	Display the error massage & terminate the program	9	Ok

SCREEN SHOTS –STUDENT A

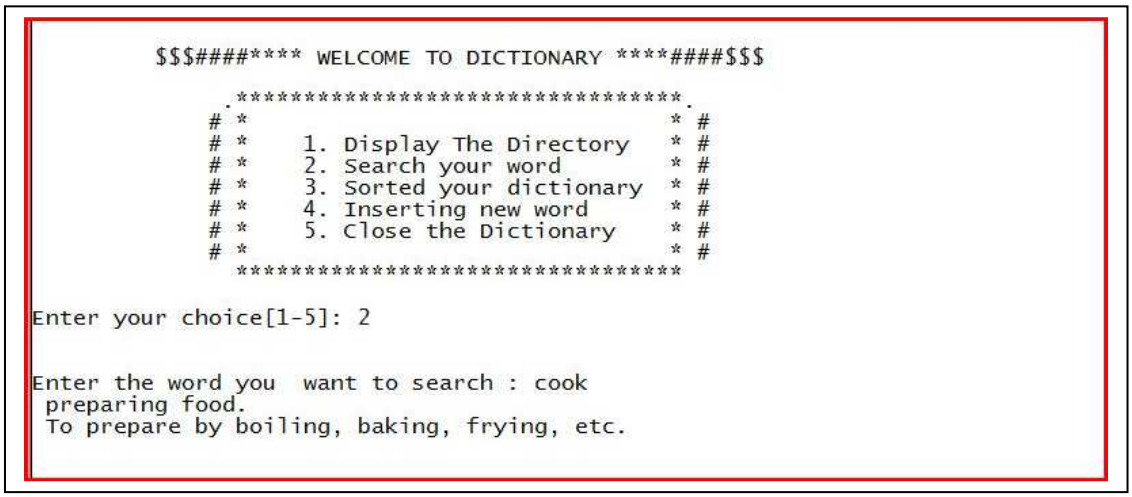
Welcome to Dictionary



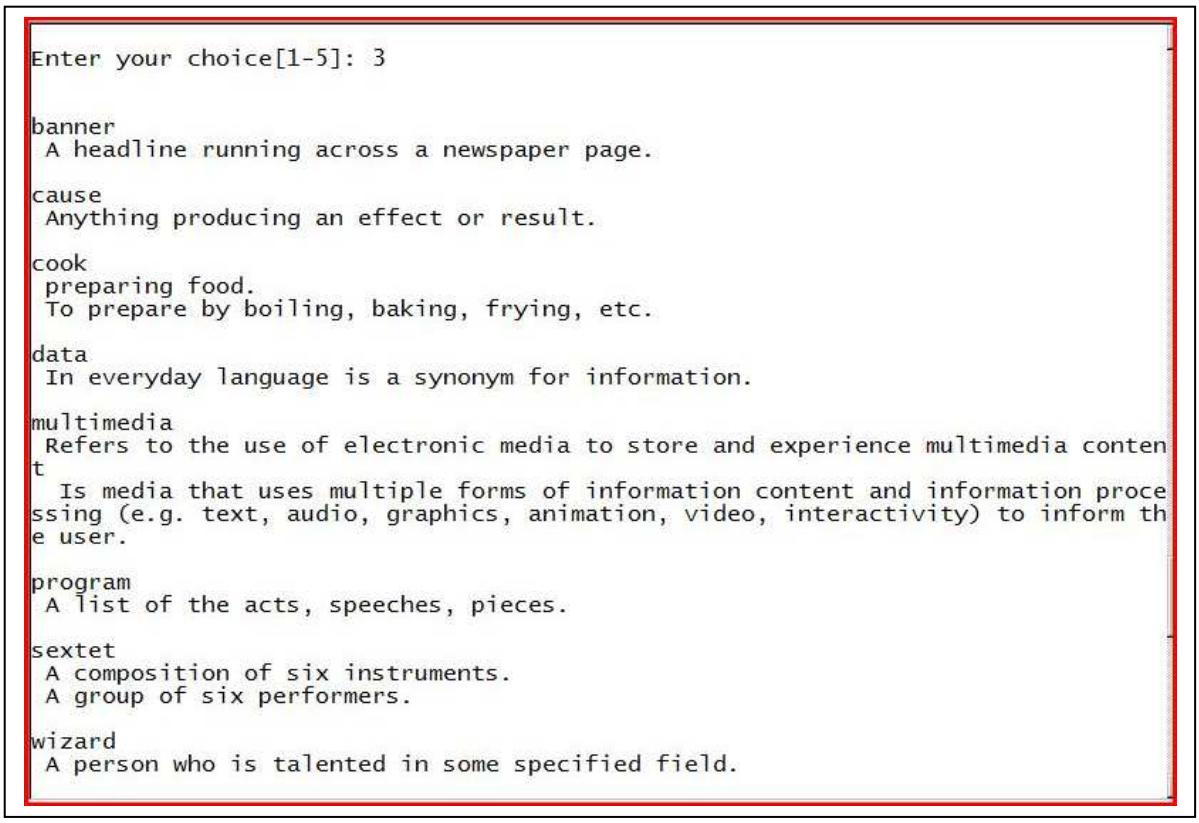
Screen shot 1



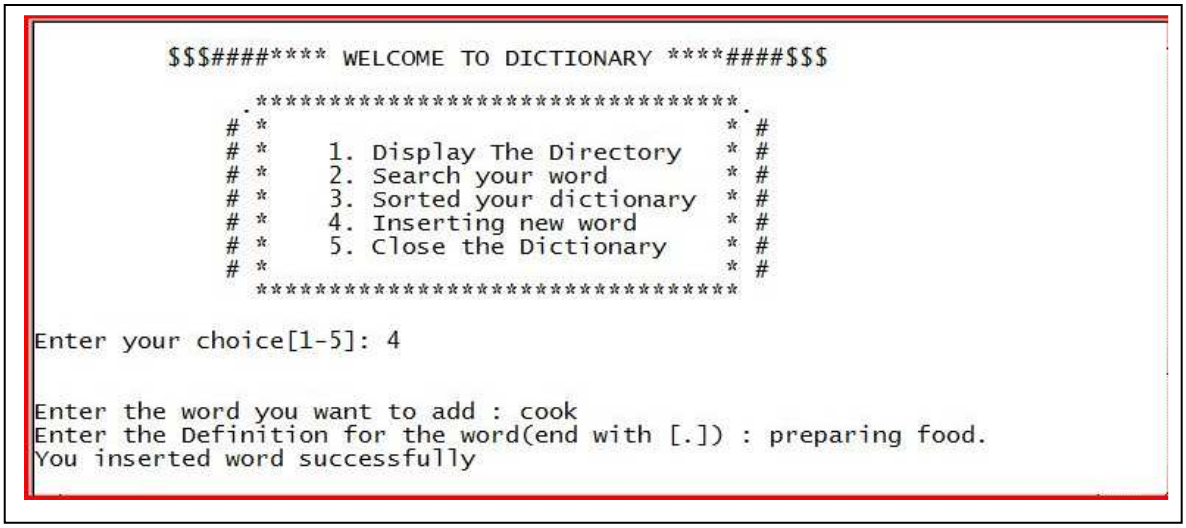
Screen shot 2



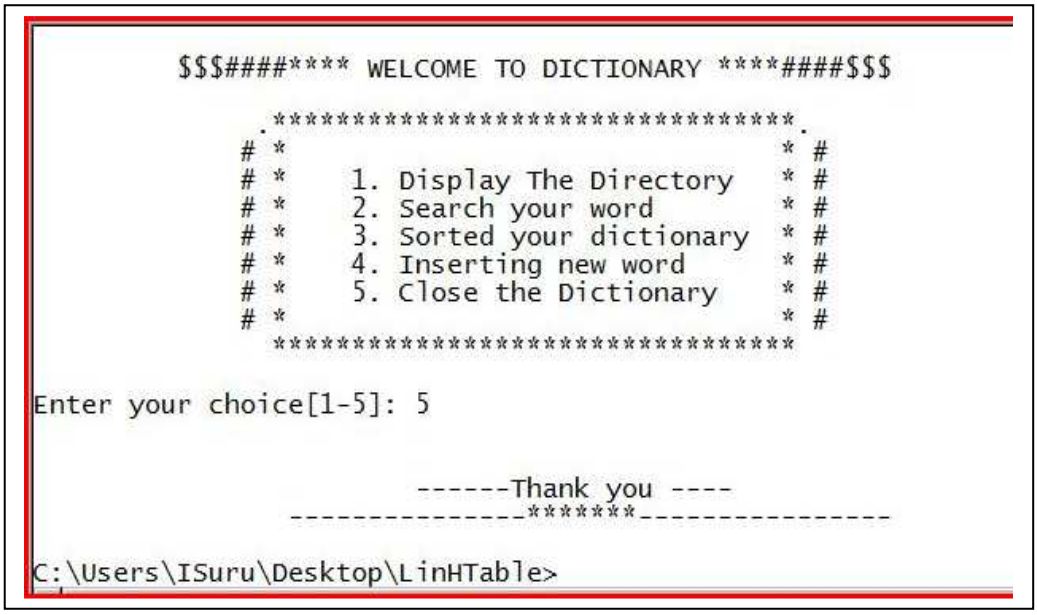
Screen shot 3



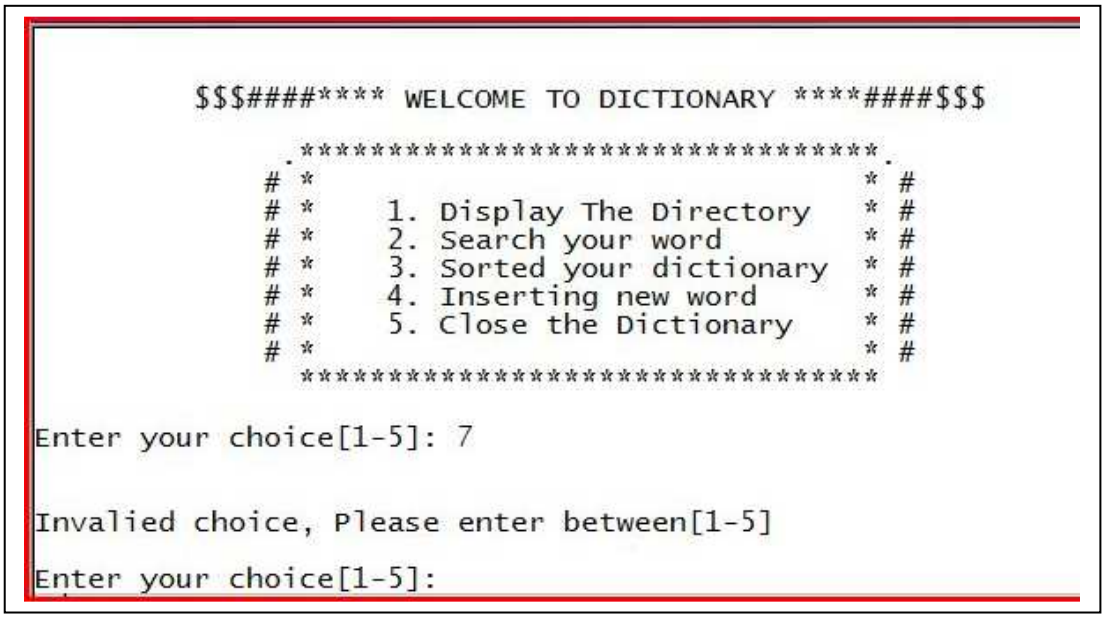
Screen shot 4



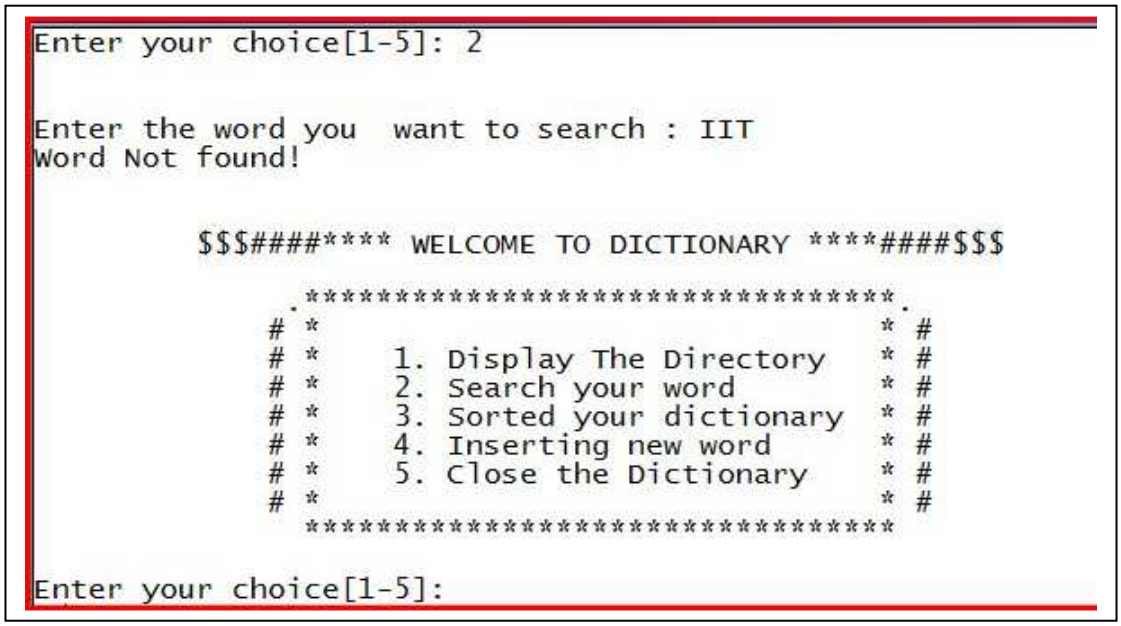
Screen shot 5



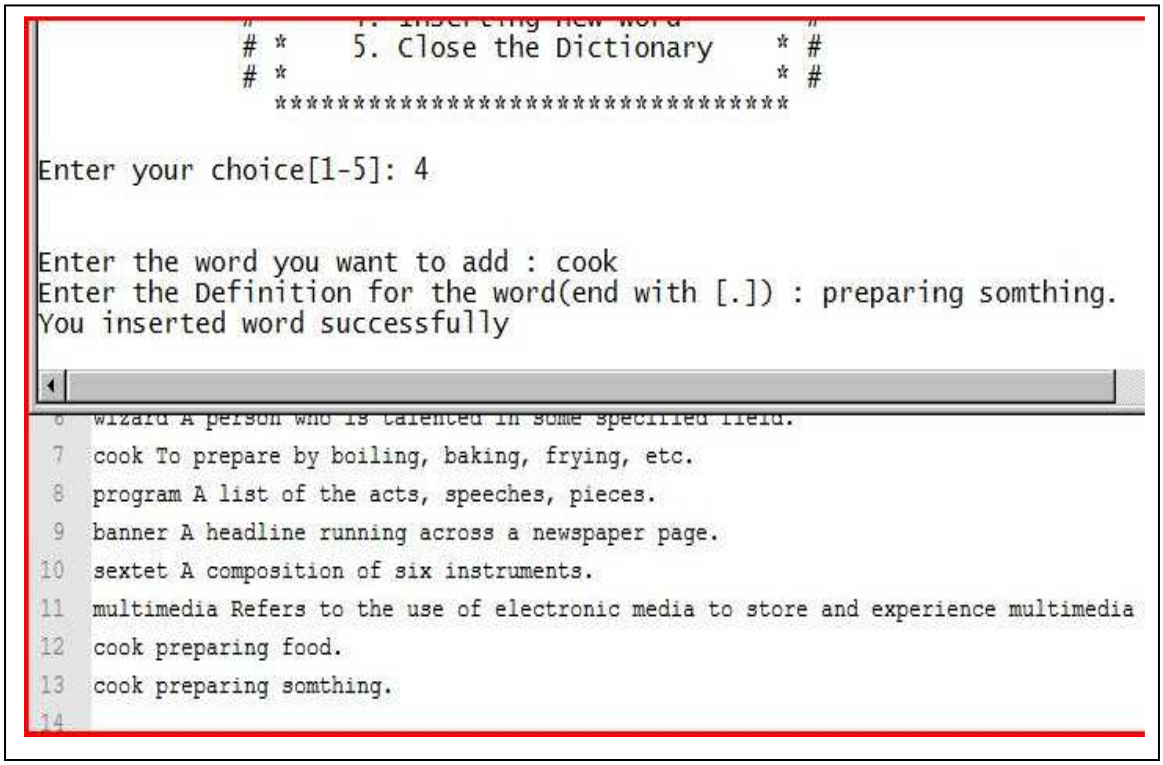
Screen shot 6



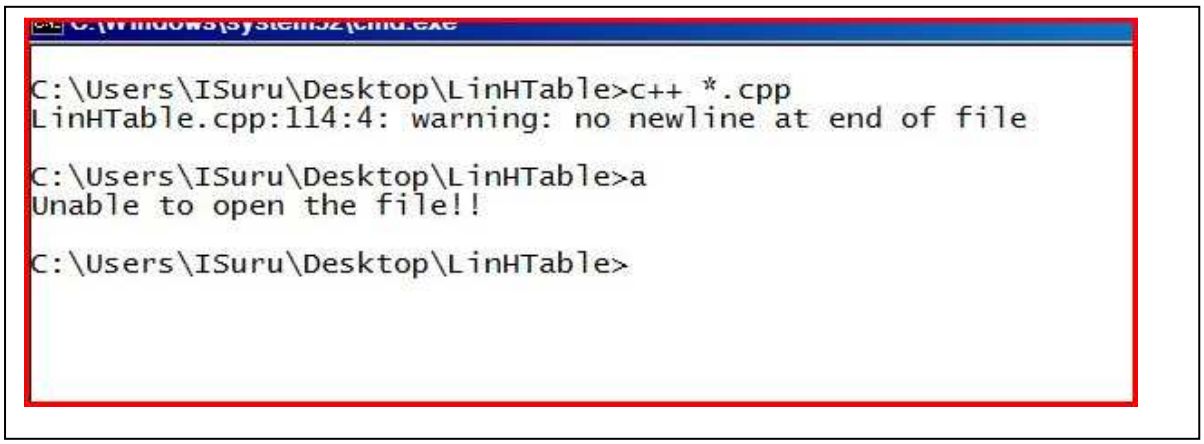
Screen shot 7



Screen shot 8



Screen shot 9

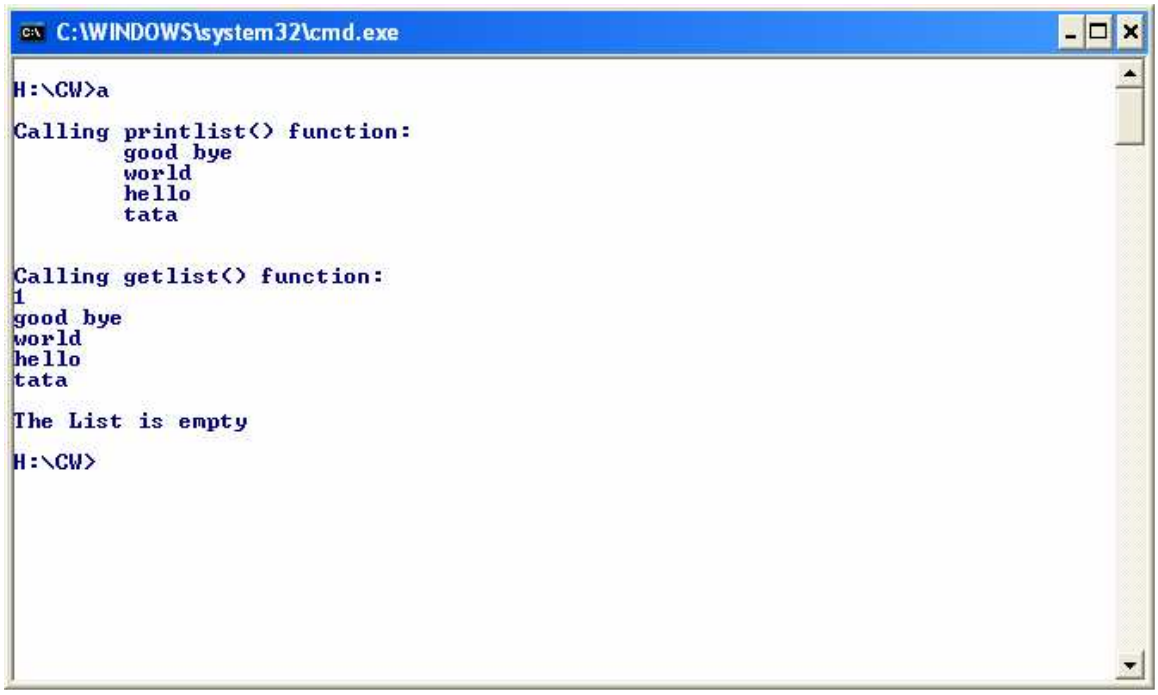


Student B

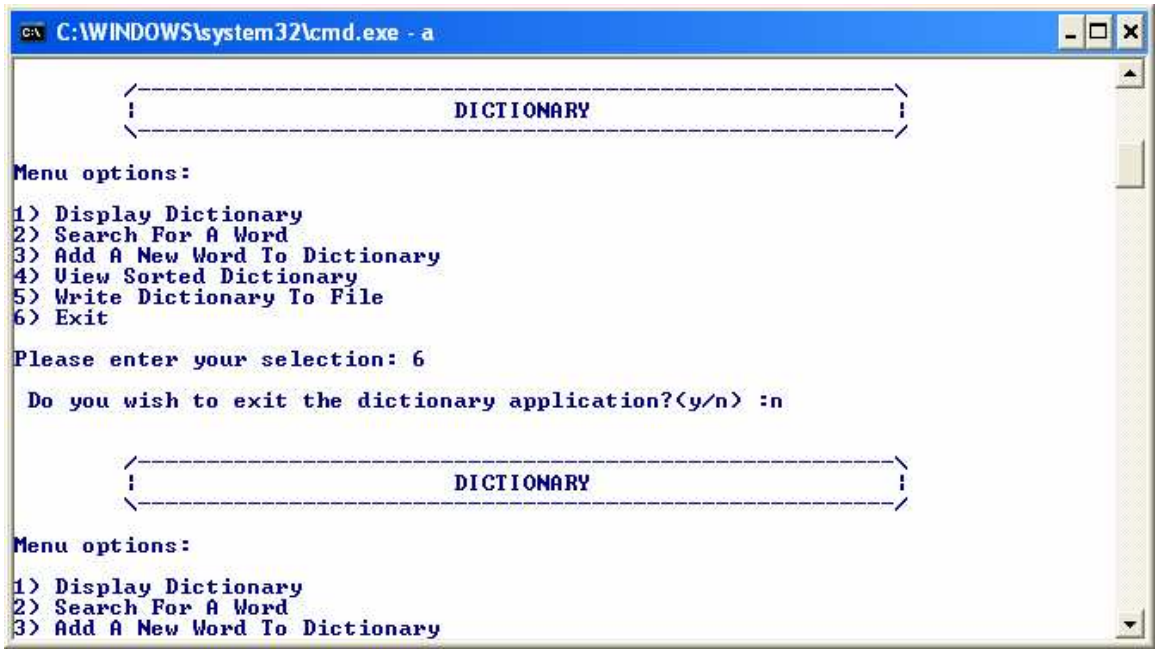
Test No	Test	Expected out put	Actual output
1	Test getList() function in the List class	List is displayed	List is displayed in a similar manner to printList() function
Testing Indep_test.cc			
2	Check if menu is displayed continuously after each operation	Menu should be shown after each operation	Menu is displayed
3	Enter an invalid number as menu option	Displays error message	Displays error message. Asks user if application is to be closed
4	Exit from application	Exits	User is informed that application is being exited from and ends program
5	Test menu option 1	Prints out dictionary on screen	Dictionary is being displayed
6	Test menu option 2	Conducts a search and displays results	Prompts user for word to search for and displays results
7	Test menu option 2 with invalid word	Conducts a search and displays no results	Displays error message that word hasn't been found
8	Test menu option 2 with a word with an invalid character	Displays error message	Displays error message that word is not valid
9	Test menu option 3	Word is added	Word is added to the defs.dat file and will be included the next application runs.
10	Test menu option 3 with a word with invalid characters	Displays error message	Displays error message
11	Test menu option 4	Display sorted dictionary	Display sorted dictionary
12	Test menu option 5	Writes to file in the given format	Writes to file and displays message to user confirming it

SCREEN SHOTS – STUDENT B

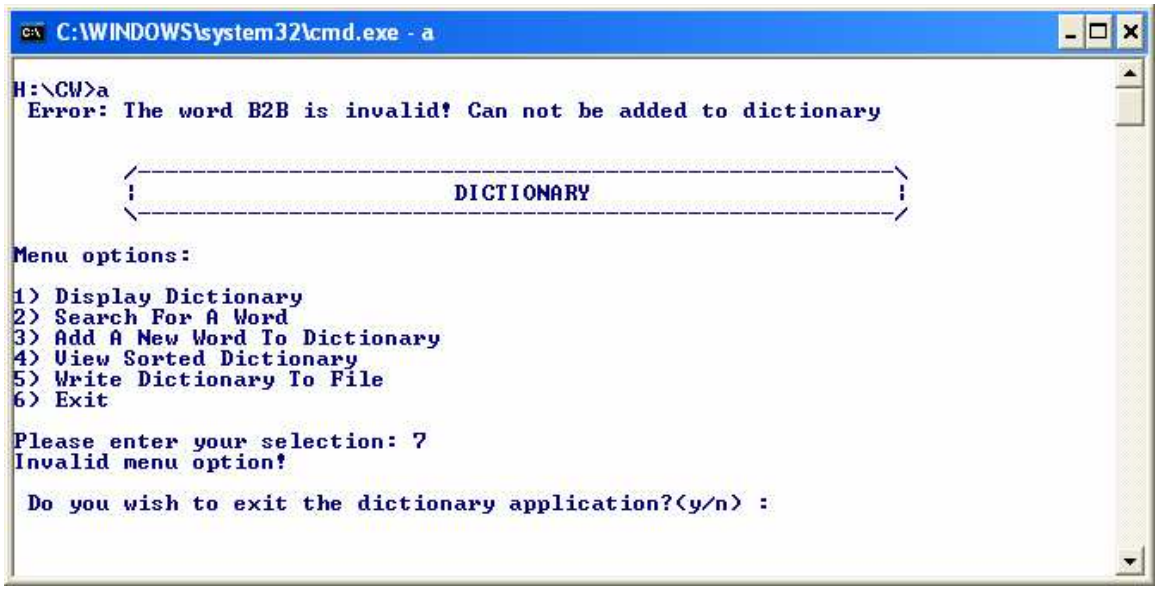
Test 1



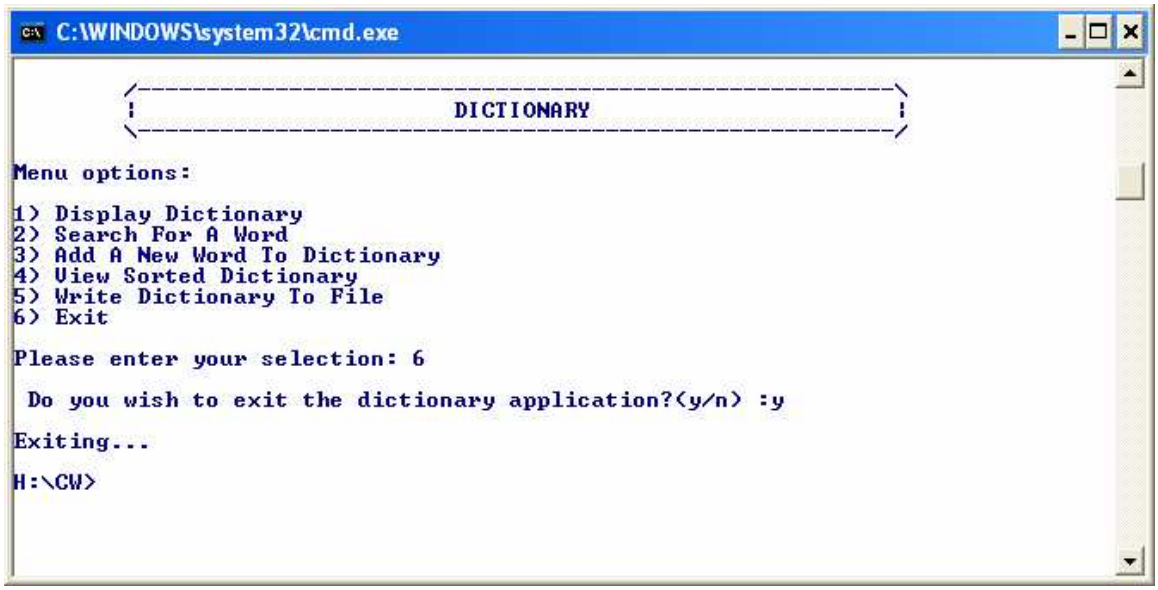
Test 2



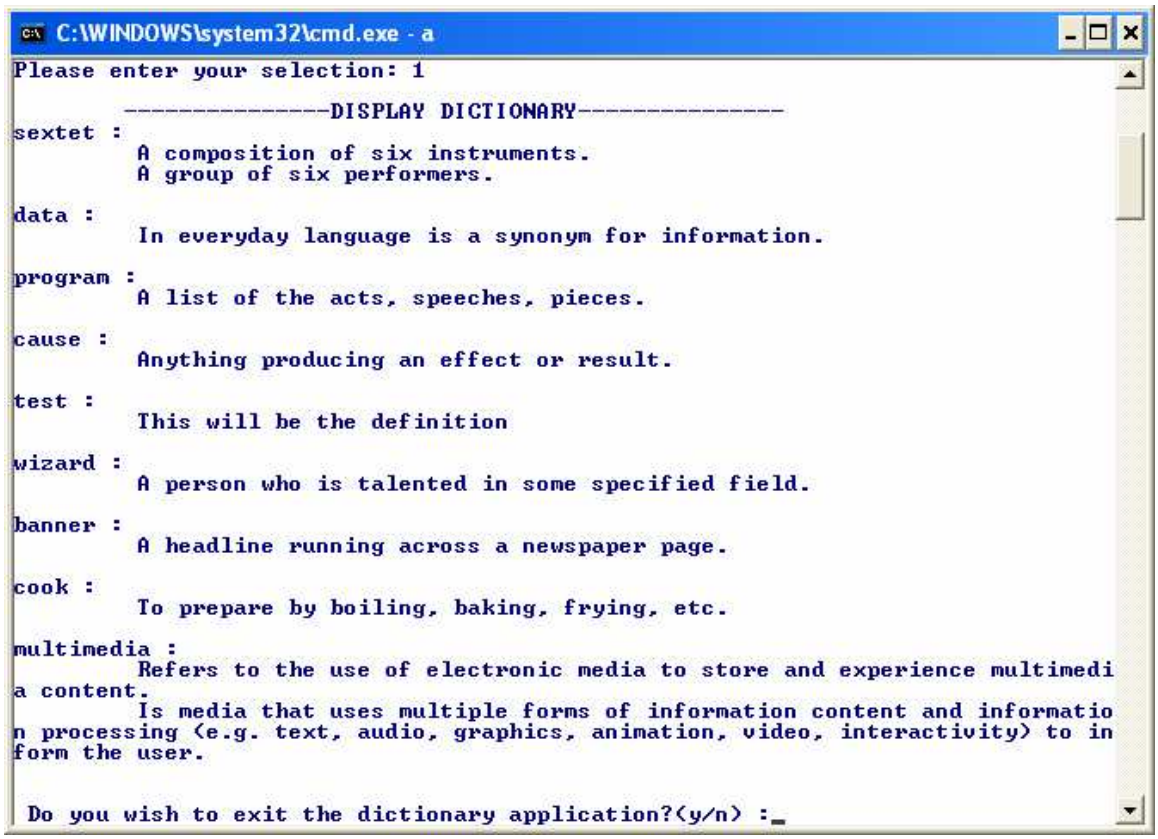
Test 3


C:\WINDOWS\system32\cmd.exe - a
H:\CW>a
Error: The word B2B is invalid! Can not be added to dictionary
DICTIONARY
Menu options:
1) Display Dictionary
2) Search For A Word
3) Add A New Word To Dictionary
4) View Sorted Dictionary
5) Write Dictionary To File
6) Exit
Please enter your selection: 7
Invalid menu option!
Do you wish to exit the dictionary application?(y/n) :

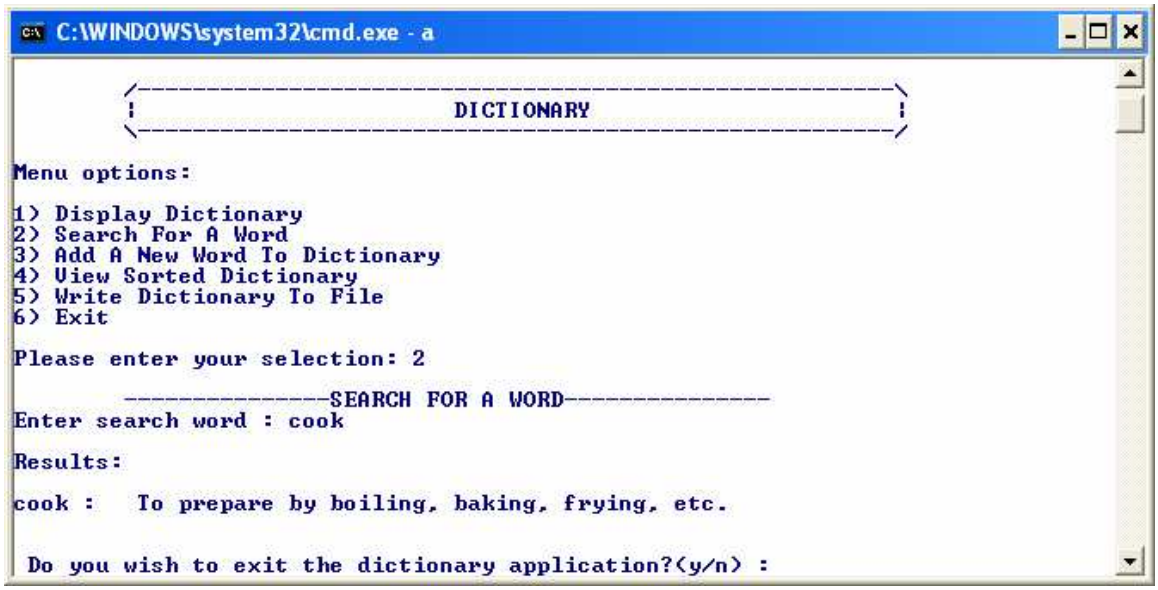
Test 4


C:\WINDOWS\system32\cmd.exe
DICTIONARY
Menu options:
1) Display Dictionary
2) Search For A Word
3) Add A New Word To Dictionary
4) View Sorted Dictionary
5) Write Dictionary To File
6) Exit
Please enter your selection: 6
Do you wish to exit the dictionary application?(y/n) :y
Exiting...
H:\CW>

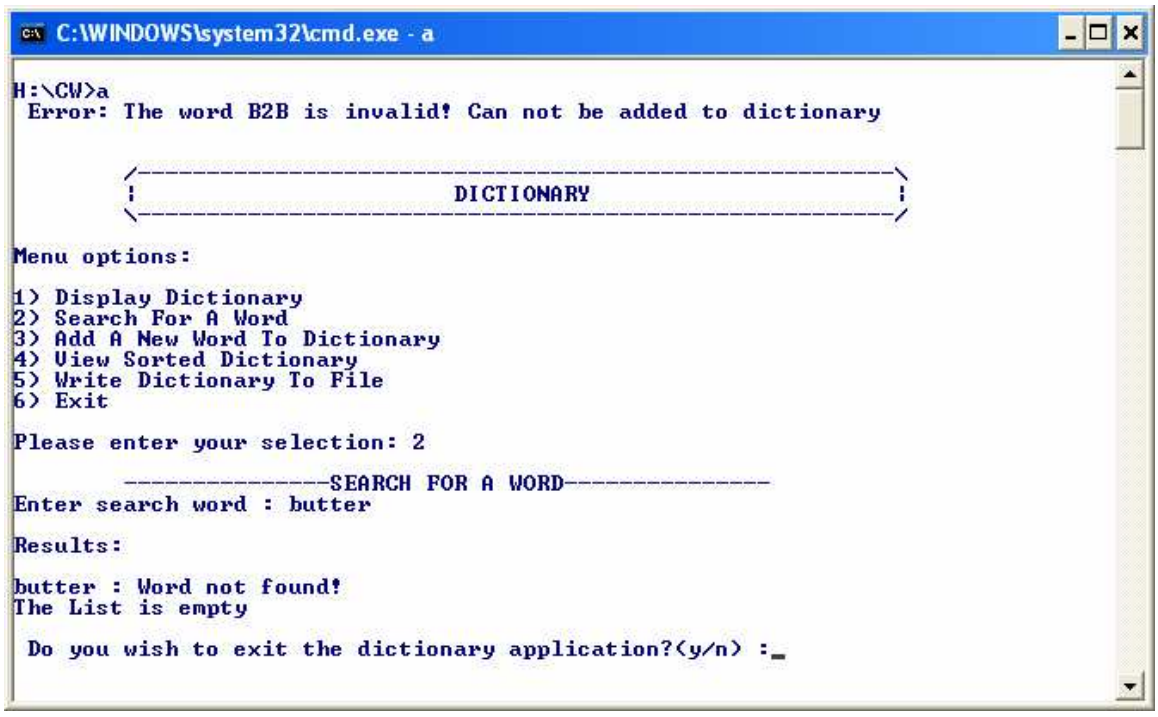
Test 5


C:\WINDOWS\system32\cmd.exe - a
Please enter your selection: 1
----------DISPLAY DICTIONARY----------
sextet :
A composition of six instruments.
A group of six performers.
data :
In everyday language is a synonym for information.
program :
A list of the acts, speeches, pieces.
cause :
Anything producing an effect or result.
test :
This will be the definition
wizard :
A person who is talented in some specified field.
banner :
A headline running across a newspaper page.
cook :
To prepare by boiling, baking, frying, etc.
multimedia :
Refers to the use of electronic media to store and experience multimedi
a content.
Is media that uses multiple forms of information content and informatio
n processing (e.g. text, audio, graphics, animation, video, interactivity) to in
form the user.
Do you wish to exit the dictionary application?(y/n) :

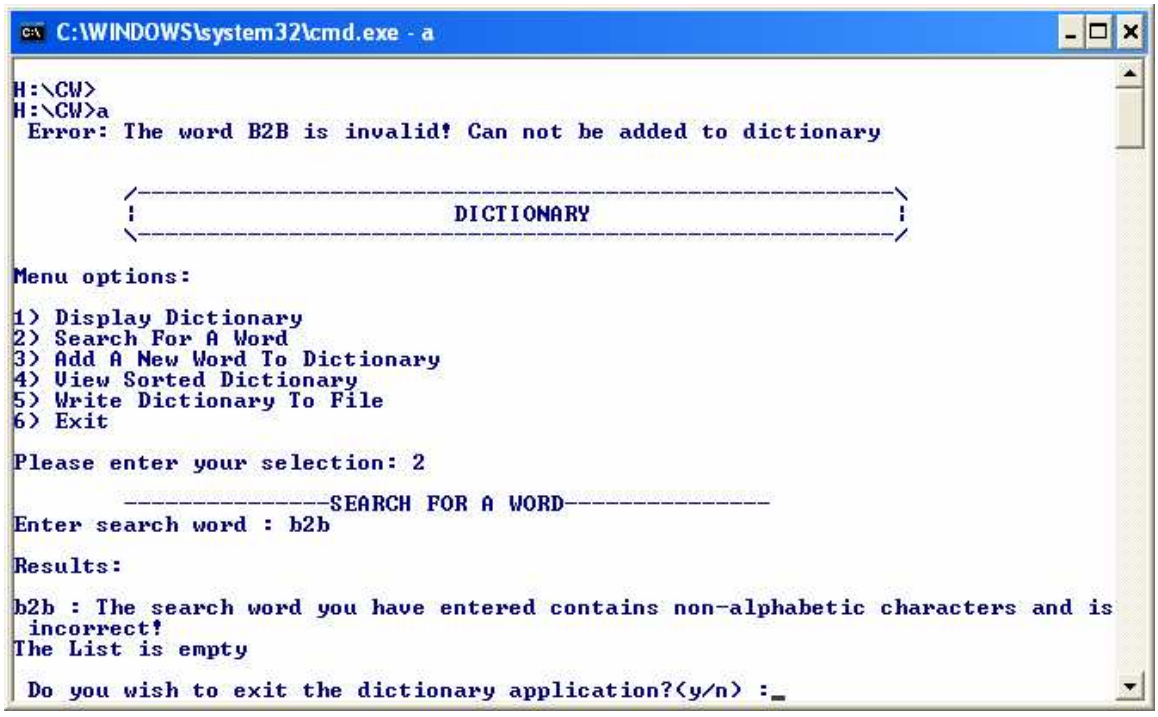
Test 6


C:\WINDOWS\system32\cmd.exe - a
DICTIONARY
Menu options:
1) Display Dictionary
2) Search For A Word
3) Add A New Word To Dictionary
4) View Sorted Dictionary
5) Write Dictionary To File
6) Exit
Please enter your selection: 2
----------SEARCH FOR A WORD----------
Enter search word : cook
Results:
cook : To prepare by boiling, baking, frying, etc.
Do you wish to exit the dictionary application?(y/n) :

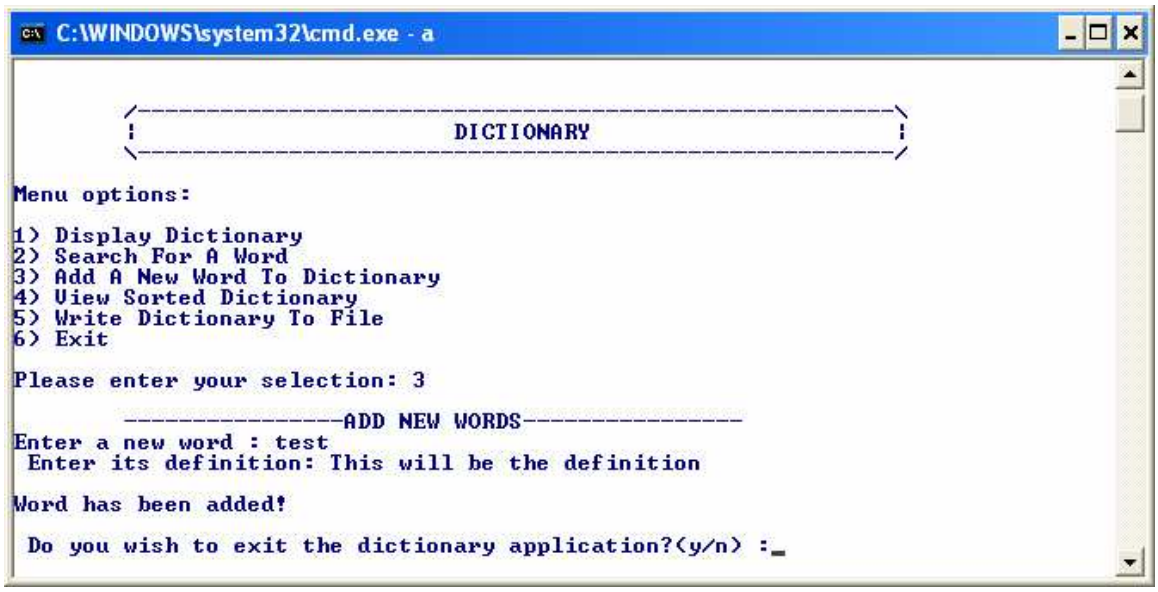
Test 7


C:\WINDOWS\system32\cmd.exe - a
H:\CW>a
Error: The word B2B is invalid! Can not be added to dictionary
DICTIONARY
Menu options:
1) Display Dictionary
2) Search For A Word
3) Add A New Word To Dictionary
4) View Sorted Dictionary
5) Write Dictionary To File
6) Exit
Please enter your selection: 2
SEARCH FOR A WORD
Enter search word : butter
Results:
butter : Word not found!
The List is empty
Do you wish to exit the dictionary application?(y/n) :

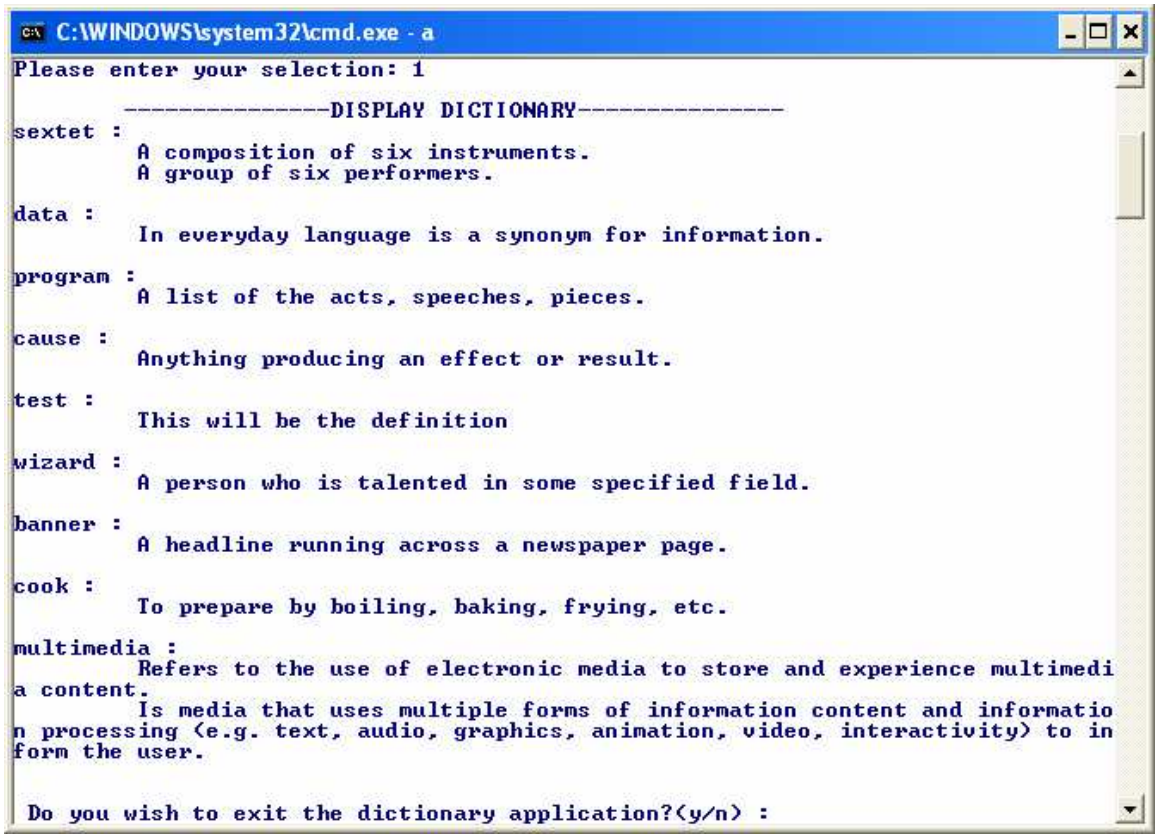
Test 8


C:\WINDOWS\system32\cmd.exe - a
H:\CW>
H:\CW>a
Error: The word B2B is invalid! Can not be added to dictionary
DICTIONARY
Menu options:
1) Display Dictionary
2) Search For A Word
3) Add A New Word To Dictionary
4) View Sorted Dictionary
5) Write Dictionary To File
6) Exit
Please enter your selection: 2
SEARCH FOR A WORD
Enter search word : b2b
Results:
b2b : The search word you have entered contains non-alphabetic characters and is incorrect!
The List is empty
Do you wish to exit the dictionary application?(y/n) :

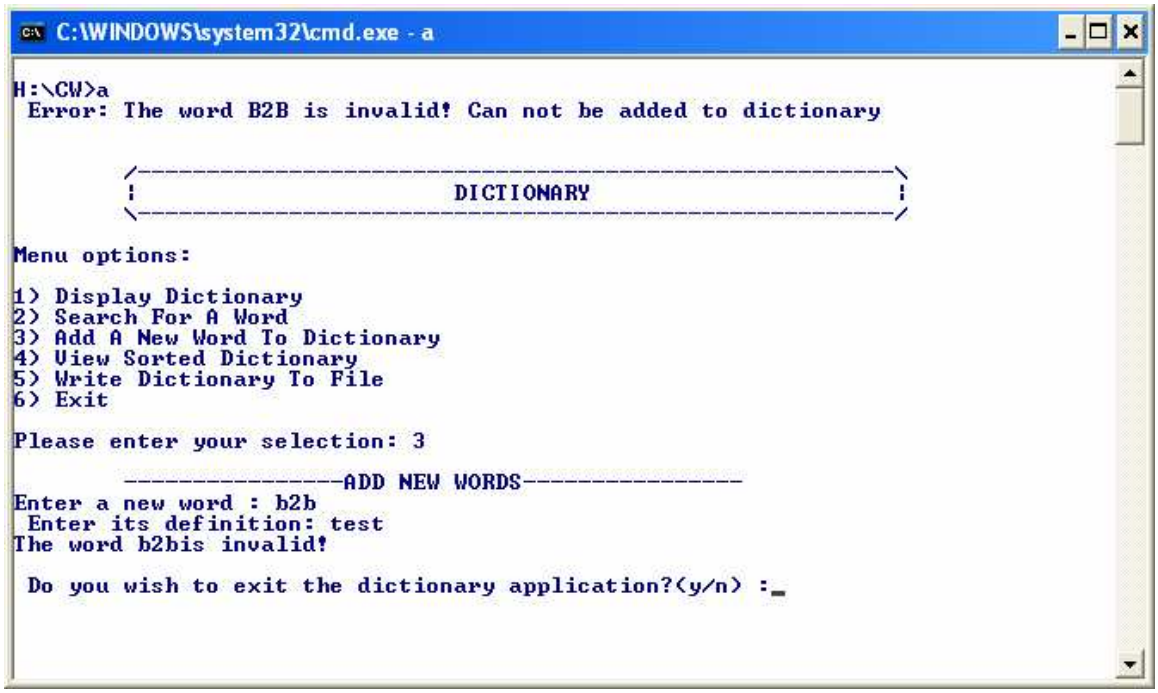
Test 9



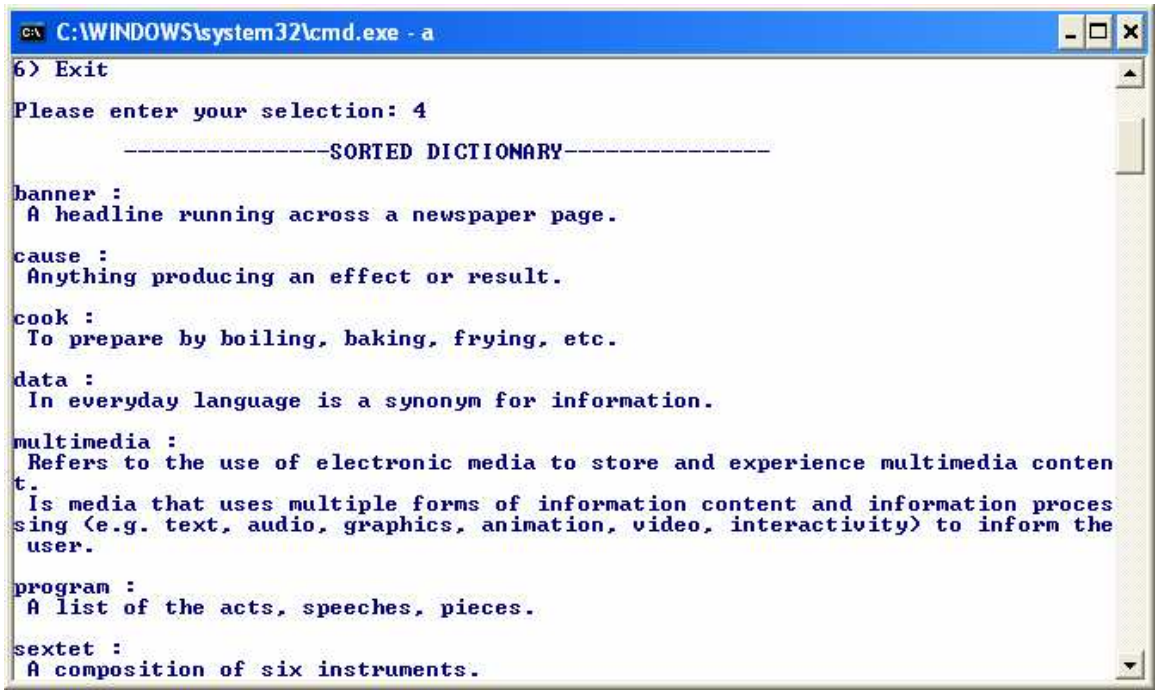
In order to show that the word has been added, I displayed the dictionary in the next screenshot



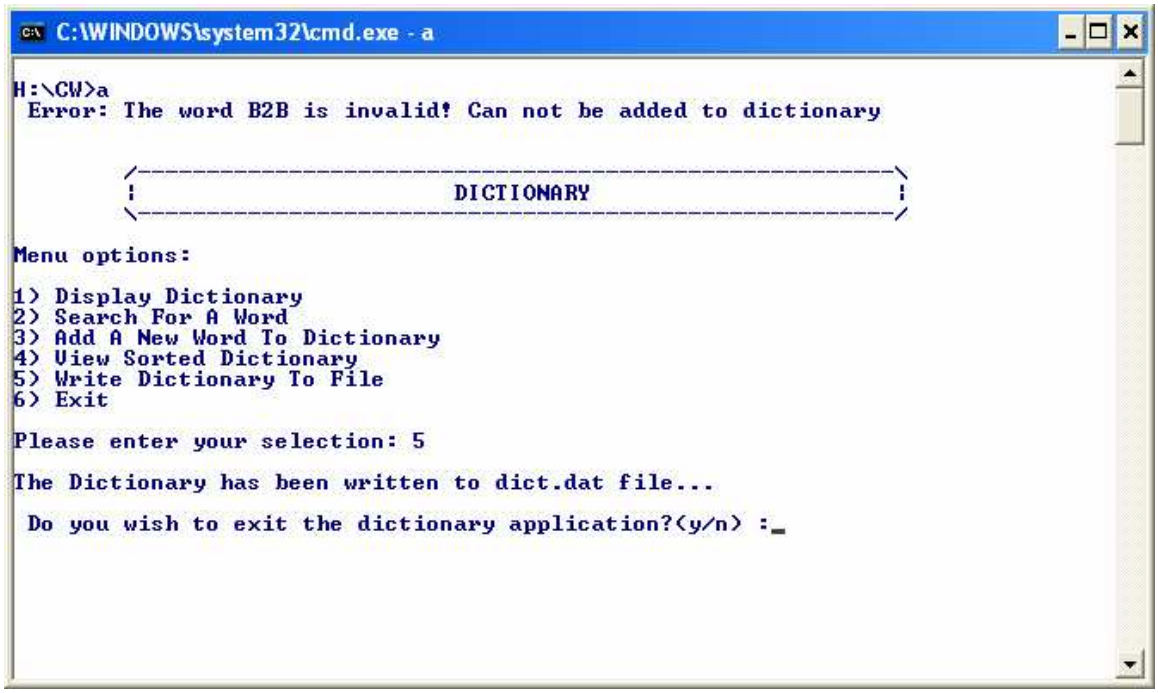
Test 10

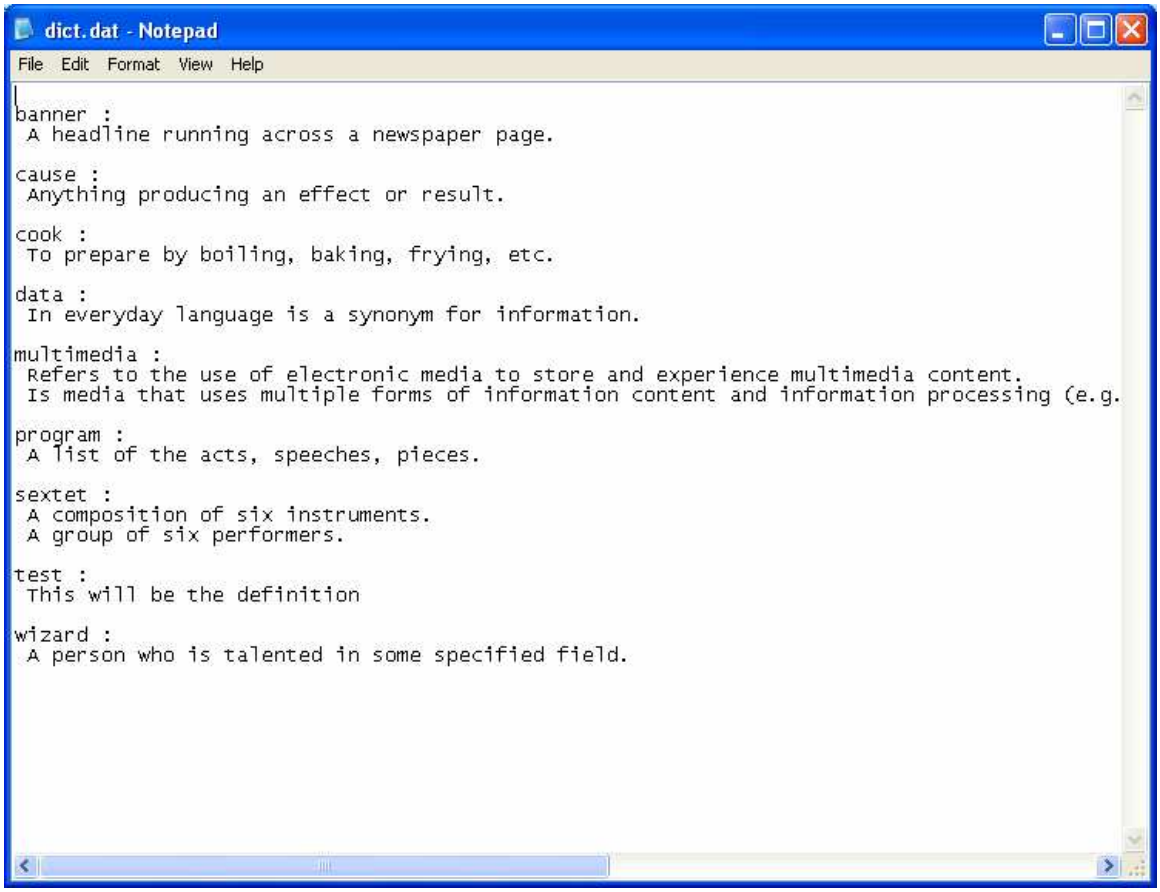


Test 11



Test 12


C:\WINDOWS\system32\cmd.exe - a
H:\CW>a
Error: The word B2B is invalid! Can not be added to dictionary
DICTIONARY
Menu options:
1) Display Dictionary
2) Search For A Word
3) Add A New Word To Dictionary
4) View Sorted Dictionary
5) Write Dictionary To File
6) Exit
Please enter your selection: 5
The Dictionary has been written to dict.dat file...
Do you wish to exit the dictionary application?(y/n) :


dict.dat - Notepad
File Edit Format View Help
banner :
A headline running across a newspaper page.
cause :
Anything producing an effect or result.
cook :
To prepare by boiling, baking, frying, etc.
data :
In everyday language is a synonym for information.
multimedia :
Refers to the use of electronic media to store and experience multimedia content.
Is media that uses multiple forms of information content and information processing (e.g.
program :
A list of the acts, speeches, pieces.
sextet :
A composition of six instruments.
A group of six performers.
test :
This will be the definition
wizard :
A person who is talented in some specified field.

EVALUATION

Individual Statements- Self-Evaluation

Student A

Student Name:	**N.T.I.Kavindya**	***Student Id:***	**2007053**

State what tasks you carried out in the project.

- My task is implementing about linear hash table (Student A).
- I worked on the Definition and List classes with my partner. We added new methods when needed and implemented them.
- I implemented the Linear Hash Table.
- I Created algorithm to convert words to their numeric representation by using reference books.

State what you enjoyed and did not enjoy about teamwork.

I enjoyed working with a group member. It helped us to come up with fresh ideas &
We were able to share our knowledge.

State what you learnt about teamwork.

The work done together as a team is greater than the sum of work done individually.

State what skills you gained/learnt from undertaking the project.

I learnt how a link list works and how to use a hash table.
And also I could understand about pointers.

State any strength about yourself that emerged whilst undertaking the project.

Understanding the scenario and writing code for it.

State any weaknesses about yourself that emerged whilst undertaking the project.

The weakness that I waste time to think & I don't satisfy about my time management.

State how you would do things better if you were to undertake the project again.

We will do our project to the best.

Has undertaking this module made you reflect on/change your course.

Ofcourse, It will Help us to learn how to work in a team environment and
It has helped me understand how some of the standard library features work.

Additional general or project specific comments:

We did the best with the time given to us.

Student Signature:		***Date:***	14-11-2008

Student B

Student Name:	**Malshani Nanayakkara**	***Student Id:***	**2007020**

State what tasks you carried out in the project.

As a group, we implemented the Definition and List classes, as well as the test class for them. Individually, I implemented the independent hash table.

State what you enjoyed and did not enjoy about teamwork.

Working as a team, we helped each other. But I found a bit discouraging was the fact I found it hard to communicate with my teammate. But in the end, I think we worked through it.

State what you learnt about teamwork.

Working in a team can help you where working individually can not. In a team, you can support and help one another, and the other teammates will help you in return.

State what skills you gained/learnt from undertaking the project.

I learnt about the linked list and how to work with objects and classes. This project required us to be thorough in these two fields, so I now understand how they work.

State any strength about yourself that emerged whilst undertaking the project.

Patience.. It takes a lot of patience to sit and work out a complicated bug and think of an alternative if it still doesn't work out. Keeping a person's cool in a troubled time is also an added bonus.

State any weaknesses about yourself that emerged whilst undertaking the project.

Time management. I feel like I should have not delayed working on this project during times where I could have, as I feel like I have rushed through towards the end.

State how you would do things better if you were to undertake the project again.

We could have made a better interface and added extra functions.

Has undertaking this module made you reflect on/change your course.

Nope.

Additional general or project specific comments:

Student Signature:		***Date:***	14-11-2008