**Question 1**

1.1 happy\_string[6] = ‘Z’ ;cannot insert into the null terminator,

1.2 C strings use **strcpy(happy\_string, “DobeDO”);**

Copies string **“DobeDO”** into string happy\_string.

1.3 use strcmp(happy\_string,sad\_string) for C-string

If strcmp(! (happy\_string,sad\_string ) )

Cout << “ The strings are the same “;

**Question 2**

2.1 line2: #include <casset>

line7: assert(term > 0);

2.2 if (term ==1) return 0;

2.3 return first + dif \* (term-1);

**Question 4**

4.1

#ifndef ENTRY\_H

#define ENTRY\_H

#include <iostream>

#include <string>

using namespace std;

class Entry

{

public:

Entry();

Entry( string n,string ad,string no);

string get\_name();

string get\_address();

string get\_number();

void update(string ad, string no);

friend bool operator == (const Entry& entry1,const Entry& entry2);

friend ostream& operator << (ostream& outs, const Entry& e );

friend istream& operator >> (istream& ins, Entry& e );

private:

string name,address,number;

};

#endif // ENTRY\_H

4.2

#include "Entry.h"

#include <iostream>

#include <string>

using namespace std;

Entry::Entry()

{

name = "";

address = "";

number = "";

}

Entry::Entry( string n,string ad,string no){

name = n;

address = ad;

number = no;

}

string Entry::get\_name(){

return name;

}

string Entry::get\_address(){

return address;

}

string Entry::get\_number(){

return number;

}

void Entry::update(string ad, string no){

address = ad;

number = no;

}

bool operator == (const Entry& entry1,const Entry& entry2){

if(entry1.name==entry2.name && entry1.address==entry2.address && entry1.number==entry2.number)

return true;

else

return false;

}

ostream& operator << (ostream& outs, const Entry& e ){

outs << e.name << "\t" << e.address << "\t" << e.number <<endl;

return outs;

}

istream& operator>> (istream& ins, Entry& e ){

getline(ins,e.name,'\n');

getline(ins,e.address,'\n');

ins >> e.number;

return ins;

}

4.3

#include "Entry.h"

#include <iostream>

#include <string>

using namespace std;

int main()

{

Entry Entry1("David Allan","5 Flowers street, Lynnwood Ridge, Pretoria","(012)989889");

Entry Entry2;

cin >> Entry2;

if(Entry1==Entry2)

cout << "You entered a duplicate value!" << endl;

Entry1.update("12 Conifer street,Morningside, Durban", "(031)555777");

return 0;

**Question 5**

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

int main()

{

char out\_file\_name[16];

ifstream fin;

ofstream fout;

cout << "Enter the output file name (maximum of 15 characters):\n";

cin >> out\_file\_name;

fin.open("CodeMe.txt");

if(fin.fail( ))

{

cout << "Input file opening failed.\n";

exit(1);

}

fout.open(out\_file\_name);

if (fout.fail( ))

{

cout << "Output file opening failed.\n";

exit(1);

}

char ch ;

while (!fin.eof()) {

fin.get(ch);

if(!isspace(ch)){

ch =ch + 10;

}

fout << ch;

}

fin.close();

fout.close();

return 0;

}

**Question 6**

6.1

#ifndef MUSICCOMPETITOR\_H

#define MUSICCOMPETITOR\_H

#include <iostream>

using namespace std;

class MusicCompetitor :Public Competitor

{

public:

void get\_instrument(string i);

void calc\_final\_mark() const;

private:

string name;

string competitor\_ID;

string item;

int marks[5];

string instrument;

};

6.2

MusicCompetitor:: MusicCompetitor(string new\_name,string new\_ID, string new\_item, string new\_instrument)

: Competitor(string new\_name, string new\_ID, string new\_item),instrument(new\_intrument){}

6.3.1 The program is accessing final\_mark and marks[5] from the base class Competitor.

6.3.2 Declare them again in MusicCompetitor or make them public in Competitor.Declare like in Q6.1 or make them public variables.

**Question 7**

Template <class TStudent, class TContact>

Class StudentDetail {

Public:

StudentDetail();

Void Add ( TStudent student, const TContact& contact);

TContat Lookup(TStudent student) const;

Private:

vector <TStudent> Student;

vector <TContact> Contact;

};

7.2

Template <class TStudent, class TContact>

Void StudentDetail< TStudent, TContact >::Add( TStudent student, const TContact& contact){

Student.push\_back(student);

Contact.push\_back(contact);

}

7.3

StudentDetail <string, double> student1();