

**COS1512
RCO1512**

October/November 2015

INTRODUCTION TO PROGRAMMING II

Duration 2 Hours

75 Marks

EXAMINERS

FIRST

SECOND

MRS HW DU PLESSIS

PROF ID SANDERS

MRS MA SCHOEMAN

Closed book examination

This examination question paper remains the property of the University of South Africa and may not be removed from the examination venue

This paper consists of 7 pages and 7 questions.
Please make sure that you have all 7 pages with the 7 questions.

Instructions / Instruksies:

- Answer all the questions
- Do all rough work in the answer book
- The mark for each question is given in brackets next to the question
- Please answer the questions in the correct order. If you want to do a question later, leave enough space
- Number your answers and label your rough work clearly
- Marks are awarded for part of an answer, so do whatever you are able to in each question

GOOD LUCK!

[TURN OVER]

QUESTION 1**[4]**

Explain why each of the following statements is problematic

```
1 1 char happy_string[7] = "DoBeDo",
    happy_string[6] = 'Z',
```

 (1)

```
1 2 char happy_string[7],
    happy_string = "DoBeDo",
```

 (1)

```
1 3 char happy_string[7] = "happy",
    char sad_string[7] = "sad",
    if (happy_string == sad_string)
        cout<<"The strings are the same",
```

 (2)
QUESTION 2**[4]**

An arithmetic series is a number of integers in ascending or descending order where the difference between any two adjacent terms is a constant, for example 2, 6, 10, 14, 18, 22, 26, 30 is an arithmetic series in ascending order of which 2 is the first term, 22 is the 6th term and the constant difference is 4

The following incomplete program uses recursion to calculate the nth term of an ascending arithmetic series starting at 1, with a constant difference of 3, i.e. the series will be 1, 4, 7, 10, 13, 16, 19, etc

Answer the questions below and provide code for the lines indicated **Do not copy the complete program in your answer book** Only write down the line number and the answer

```
1 #include <iostream>
2.
3 using namespace std,
4 int arith( int term )
5 {
6 int first = 1, int diff = 3,
7.
8     if (term == 1)
9         return ( 1),
10 else
11.
12 }
13 int main()
14 {
15     int term, answer,
16     cout << "Please key in the number of the term " << endl,
17     cin >> term ,
18     answer = arith(term),
```

[TURN OVER]

```

19.  cout << "the value of term " << term << " is " << answer ;
20    return 0;
21 }

```

2 1 Ensure that the value for term is greater than 0, by using the assert statement. Write down only the line number where the assert statement should be inserted, with the correct version of the assert statement next to it. Also remember to include the correct #include directive in line 2. (2)

2 2 What is the base case? (1)

2 3 Determine what the general case should be and add the code in line 11. (1)

QUESTION 3

[5]

3 1 Consider the following code fragment which is assumed to be embedded in a complete and correct C++ program

```

1    int *p,
2    int *q;
3    p = new int,
4    *p = 43;
5    q = p;
6    *q = 52;
7    p = new int,
8    *p = 78;
9    q = new int;
10   *q = *p;

```

Show diagrammatically the state of memory after

3 1 lines 1-5 have been executed (2)

3 2 lines 6-10 have been executed (3)

QUESTION 4

[27]

Use separate compilation to define a class called Entry whose instance represents a single entry in a telephone directory

The class has the following member variables

name (string)	- person's name	eg Jones Tom
address (string)	- person's address	eg 51 Rissik Street, SunnySide
number (string)	- person's number	eg (012) 555 7777

In addition, the class should contain the following member functions

[TURN OVER]

- A **default constructor** that initialises the name, address and number to empty strings respectively
- An **overloaded constructor** that accepts a new entry's name, address and number as specified by three parameters
- **Accessor functions** `get_name()`, `get_number()`, `get_address()` to return the values in an object's name, address and number member variables respectively
- A void member function `update()` which allows an update on the telephone number and address, as specified by two string parameters
- An overloaded **equality operator** `==` (implemented as a **friend** function) with the following prototype:

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

This function returns `true` if `entry1` and `entry2` have the same values and `false` otherwise

- An overloaded insertion **operator** `<<` (implemented as a **friend** function) so that it can be used to output values of type `Entry`
- An overloaded extraction **operator** `>>` (implemented as a **friend** function) so that it can be used to input values of type `Entry`

Note You will have to use the C++ predefined function `getline()` to input the name and address as they contain whitespace characters

You should attempt the solutions as follows

- 4.1 Create the header file `Entry.h` that contains the `Entry` class specification (7)
- 4.2 Create the implementation of the class `Entry` including the **friend** functions (15)
- 4.3 Demonstrate the class in a program (`main()`) by providing code for the following (5)
 - 4.3.1 Instantiate an object of type `Entry` called `Entry1` and initialize it with following values


```
"David Alan"
"5 Flower Street, Lynnwood Ridge, Pretoria"
"(012) 989889"
```
 - 4.3.2 Read in `Entry` object `Entry2` that is input by the user
 - 4.3.3 Compare `Entry1` with `Entry2` to determine if they are duplicate entries and issue a warning to the user to indicate if they are duplicate entries
 - 4.3.4 Update `Entry1`'s address and number to

"12 Conifer Street, Morningside, Durban" and "(031) 555777"

QUESTION 5**[10]**

File encryption is the science of writing the contents of a file in a secret code. Write a C++ program that encrypts the contents of a file by reading it one character at a time and adding 10 to the ASCII code of each character before writing the coded contents out to a second file. The second file will then be a version of the first file, but written in a secret code. The input for this program is the file that must be encoded, resides in file `CodeMe.txt`, and the program should obtain the name of the second file which will contain the coded version, from the user. Your program should only encode non-whitespace characters. You can use the `bool` function `isspace()` to ignore non-whitespace chars. Function `isspace()` returns true if a character passed as parameter to it is a space. If `CodeMe.txt` for example contains the following

```
What a wonderful day is today!
It is the very first day of the rest of your life!
Enjoy and take care of it
```

The resulting encoded file would have the following contents

```
ark~ k yxno|p v nkf s} ~ynkf+
S~ s} ~ro €o|f ps|}~ nkf yp ~ro |o}~ yp fy | vspo+
Oxytf kxn ~kuo mk|o yp s~8
```

Hint Remember to cast the character back to `char` once you have added 10 to the ASCII code for it

QUESTION 6**[15]**

The class `Competitor` below describes a competitor taking part in the Eisteddfod. Consider the class specification (interface) for the class

```
1 class Competitor
2 {
3 public:
4     Competitor();
5     Competitor(string new_name, string new_ID,
6                 string new_item), //marks and final mark are
7                                     //initialized to 0
8     string get_name ( ) const;
9     string get_competitor_ID ( ) const;
10    string get_item ( )const,
11    void get_marks(int m[5])const,
12    int get_final_mark ( ) const,
13    void calc_final_mark( ) const, //determine final mark for
14                                     //competitor
15 private:
16     string name,
17     string competitor_ID,
18     string item;
19     int marks[5]; //marks allocated by five judges
```

[TURN OVER]

```

17.     int final_mark, //this is a weighted average with 50% from
18         // the 1st judge and the average of the other
19         // judges for the remaining 50%
20     }

```

- 6 1 Derive a class `MusicCompetitor` from class `Competitor`. This class has an additional member variable `instrument` that holds the instrument the competitor is playing, and additional member function `get_instrument`. The class should override member function `calc_final_mark()` to determine the final mark for the competitor.

NB: Provide only the class interface (6)

- 6 2 Implement the overloaded constructor for the class `MusicCompetitor` by invoking the base class constructor (3)

- 6 3 Consider the following implementation of the overridden `calc_final_mark()` for class `MusicCompetitor`. The final mark for `MusicCompetitors` is calculated as the average of the marks allocated by the five judges.

```

MusicCompetitor calc_final_mark()
{
    int total = 0;
    for (int i = 0, i < 5, i++)
        total += marks[i];
    final_mark = int (total/5),
}

```

When compiled, this implementation produces these two errors:

```

In member function 'void MusicCompetitor::calc_final_mark()'
Error 'int Competitor: final_mark' is private
Error 'int Competitor marks[5]' is private

```

- 6 3 1 Explain the reason(s) for these two errors (1)
- 6 3 2 Explain two different ways in which to correct these errors, and show the corresponding code fragments. NB: If you adapt the class `Competitor` **do not** copy the complete class to show how you adapt it, use the line numbers and indicate changes next to it (5)

QUESTION 7**[10]**

The university keeps a directory of student numbers and email addresses. The following class allows the user to enter a student number, and then returns the email address for that student number. If the student number does not exist in the directory yet, the student number and email address can be added.

```
class StudentDetail {
public:
    StudentDetail(),
    void Add(double student, const string &contact),
    string Lookup(double student) const,
private:
    vector <double> Student,
    vector <string> Contact,
}
```

The class `StudentDetail` has the following operations

`Add()` – adds a new student number and email address to the directory

`Lookup()` – returns the email address if the student number is entered

- 7.1 Write a template version of the class `StudentDetail`. Use the template prefix `StudentDetail <TStudent, TContact>` to re-design the `StudentDetail` interface, so that the student number could for example also be of type `string`, containing the surname and initials of the student, and the email address could for example also be a cell number of type `double` instead. (5)
- 7.2 Implement the `Add()` function of the template class `StudentDetail`. (4)
- 7.3 Provide a declaration for an object of class `StudentDetail` that has a student identification of type `string` and student contact details of type `double`. (1)