Tutorial letter 104/2/2016

Introduction to Programming II

COS1512

Semester 1

School of Computing

Examination Tutorial Letter

Bar code

UNISA university of south africa

Learn without limits.

CONTENTS

1. PURPOSE OF THIS LETTER

Dear Student,

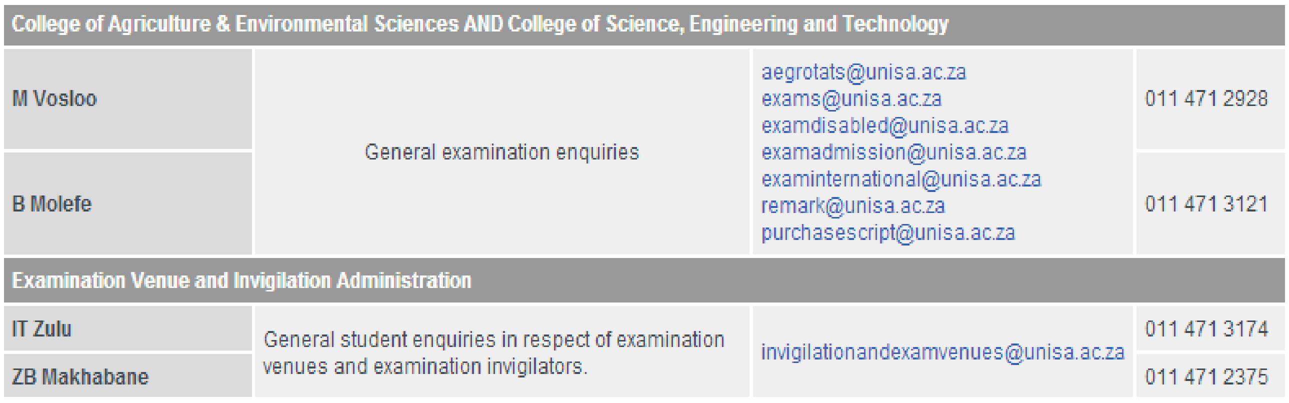
This tutorial letter contains the information pertaining to the examination as well as an examination memorandum for the May 2015 examination paper. Please make sure that you keep this letter at hand and read it often while preparing for the examination.

2. TUTORIAL MATERIAL UPLOADED

All the study material except the prescribed text book has been uploaded on our web learning portal (myUnisa). Please make sure that you visit myUnisa in order to download the material that has been uploaded. Since this module is offered mostly online, you will only receive Tutorial Letter 101 by post. Please click the following link to access myUnisa (https://my.unisa.ac.za.) and make sure that you download all the tutorial letters available on myUnisa.

3. EXAMINATION ADMISSION

The university will inform you whether you obtained examination admission or not. It will also provide you with details of the examination (date, time and the venue). It is your responsibility to make sure that the date, time and examination centre are correct. If you need any further information or details, please contact UNISA's Examination department. You can contact the Examination department on the following contact details:



College of Agriculture & Environmental Sciences AND College of Science, Engineering and Technology			
M Vosloo	General examination enquiries	aegrotats@unisa.ac.za exams@unisa.ac.za examdisabled@unisa.ac.za examadmission@unisa.ac.za examinternational@unisa.ac.za remark@unisa.ac.za purchasescript@unisa.ac.za	011 471 2928
B Molefe			011 471 3121
Examination Venue and Invigilation Administration			
IT Zulu	General student enquiries in respect of examination venues and examination invigilators.	invigilationandexamvenues@unisa.ac.za	011 471 3174
ZB Makhabane			011 471 2375

Examination admission will be based on whether or not you have submitted your assignment before the cut-off date which has been specified. If you have problems with examination admission, please make sure that you contact the Examination department. Please note that lecturers are not responsible for examination queries.

4. EXAMINATION

Please note the following:

- The exam paper covers all the work which has been outlined in tutorial letter 101 and specifically tests the outcomes specified in the tutorial letter 101. You need to use the prescribed book (Savitch 7th, 8th or 9th edition) as well as all COS1512 tutorial letters for your examination preparation.
- Work through all your assignments and tutorial letters providing solutions to the assignments. **We included additional notes in the solutions that should assist you in understanding the work.**
- Do not ignore assignment 3. Assignment 3 is self-assessment. Students frequently fail the COS1512 exam because they have not mastered the last part of the study material, which is covered by assignment 3.
- The examination consists of a single **2-hour** paper.
- The total for the examination paper is 75 marks.
- The examination is a closed book examination, i.e. you are not allowed to bring any notes or material to the examination centre.
- No calculators are allowed during the examination.
- There will be no multiple choice questions.
- The examination will be written on paper in an answer book.
- NOTE: When you work through the old exam papers, write out the answers completely. You have to practice to write it down, as you don't have a compiler in the exam to point out compiler errors.

- **Note that some questions may cover more than one learning outcome or topic.**

During the examination, ensure that you:

- Read the exam paper calmly and properly before you start answering the questions
- Read the instructions and follow them.
- Read the answers again after completing the examination.
- Complete the questions that you struggled with.

It is advisable to read the exam paper before you start answering questions. Make sure that you have completed all the answers before submitting your answer book or examination script.

5. LAYOUT OF EXAMINATION PAPER

This section provides estimates of the marks allocated per question to different sections of the work. You will need to know all the work which has been outlined in Tutorial Letter 101. The mark allocation will be approximately as follows:

Question 1: +/- 5 marks: C strings
Question 2: +/- 4 marks: recursion
Question 3: +/- 7 marks: pointers
Question 4: +/- 23 marks: class definition and implementation
Question 5: +/- 8 marks: file processing
Question 6: +/- 14 marks: inheritance
Question 6: +/- 14 marks: vectors and templates

Please note that we test all the work. This outline points out the most important aspects tested in each question.

6. PREVIOUS EXAMINATION PAPER

The examination paper which is used for this section is on myUnisa. It is the May/June 2015 examination paper. Please download the examination paper. We have only included the memorandum for the examination.

Memorandum for May 2015 Exam

QUESTION 1 **[5]**

1.1 `'\0'` indicates the end of the C-string and this statement will cause the string `yourName` to be always an empty string. √ (1)

1.2 Cannot compare two C strings with `==`, √
also cannot use `=` to copy the value of a C-string to another. √
Correct version:

```
// strcmp returns 0 (false) if string1 equals string2
if (!strcmp(yourName, name) )                        √
     strcpy(studentName, name);                      √
```

(4)

QUESTION 2 **[4]**

2.1 Output:

```
8 converted by function b() is 1000 ✓
```

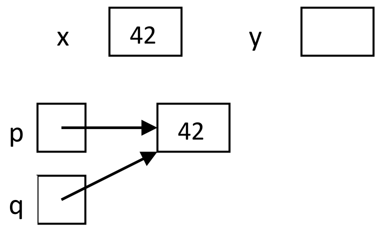
Function b() converts an integer to its binary representation ✓ (2)

2.2 and 2.3. (2)

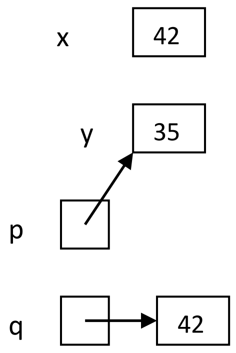
```
#include <iostream>
using namespace std;

string b (int n)     //recursive funtion
{
    string s;
    if (n%2 == 0) s = "0";
    else s = "1";
    if (n < 2) return s; //base case  ✓    (Question 2.2)
    return b (n/2) + s; //recursive case    ✓    (Question 2.3)
}

int main()
{
    string result = b (8);
    cout << "8 converted by function b() is " << result;
    return 0;
}
```

QUESTION 3 [7]

3.1.1 After lines 1-5 have been executed: √√√ (3)



3.1.2 After lines 6-7 have been executed: √√ (2)



3.2 In line 3 `q` has been de-allocated and cannot be assigned a value. (2)

QUESTION 4 [31]

Question 4.1 **(8)**

Donor.h:

```
#ifndef DONOR_H                    // for defining DONOR_H, including #endif       √
#define DONOR_H
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <iomanip>
using namespace std;

class Donor {
    public:
    friend istream& operator >> (istream& ins, Donor& the_donor);              √
    friend ostream& operator << (ostream& outs, const Donor& the_donor);       √
    friend bool operator==(const Donor & donor1, const Donor & donor2);
    Donor();                                        //default constructor      √
    Donor(string new_type);                   //overloaded constructor         √
    ~Donor();                                       //destructor               √
    string get_name()const;                   //accessor         //for all accessors   √
    string get_contact()const;             //accessor
    string get_type()const;         //accessor
    private:
        string name;                                  //for member variables            √
        string contact;
        string type;
    };

#endif //DONOR_H
```

Question 4.2 **(13)**

Donor.cpp:

```
#include <iostream>
#include <fstream>                            //  for correct  #include files           √
#include <cstdlib>
#include <iomanip>
#include "Donor.h"                                                                      √
using namespace std;

Donor::Donor()                            //default constructor
{
    name = "";                      // for initialising all member variables            √
    contact = "";
    type = "";
}

Donor::Donor(string new_name, new_contact,new_type) // overloaded constructor
{
    name = new_name;                // for initialising all member variables            √
    contact = new_contact;
    type = new_type;
 }

Donor::~Donor()                           // destructor                                 √
{

}

string Donor::get_name()const       //accessor to obtain name
{
    return name;                          //for complete accessor                       √
}

string Donor::get_contact()const      //accessor to obtain contact
{
    return contact;                       //for complete accessor                       √
}

string Donor::get_type()const       //accessor to obtain blood type
{
    return type;                          //for complete accessor                       √
}

bool operator==(const Donor & donor1, const Donor & donor2)
{
    return (donor1.type == donor2.type);                                                √
}


istream& operator >> (istream& ins,  Donor& the_donor)
```

```
{
    ins >> the_donor.name >> the_donor.contact >> the_donor.type;             √

    return ins;                                                               √
}

ostream& operator << (ostream& outs,  const Donor& the_donor)
{
    outs << the_donor.name << "\t" << the_donor.contact << "\t" << the_donor.type
   << endl;                                                                   √

    return outs;                                                              √
}
```

Question 4.3 **(10)**

Application file:

```
#include <iostream>
#include "Donor.h"                                          //for all includes   √
#include <fstream>
#include <string>
using namespace std;

int main()
{
   string blood_type;
   cout << "Enter the blood type for which the donors should be determined: ";
   cin >> blood_type;                                                         √
   Donor donors_needed(" "," ", blood_type);                                  √

   ifstream infile;                                                           √
      infile.open ("AllDonors.txt");                                          √
      if (infile.fail())
      {
         cout<<"Error opening file";
         exit(1); // for opening file"
      }

      Donor a_donor;                                                          √

   while (infile >> a_donor)                                                  √
   {
        if (a_donor == donors_needed)                                         √
            cout << a_donor;                                                  √
   }

   infile.close();                                                            √
   return 0;
 }
```

AllDonors.txt:

```
Peter 0829797987 O+
John 8769090324 O-
Sarah 3452313344 AB
Eleanor 4563423456 O+
Elaine 6784545321 O+
Erika 8765656389 O-
```

Output:

```
Enter the blood type for which the donors should be determined: O+
Peter   0829797987      O+
Eleanor 4563423456      O+
Elaine  6784545321      O+
3 donors with blood type O+ were found.Press any key to continue . . .
```

QUESTION 5 [15]

Question 5.1 **(6)**

```
class Book                                                              √
{
public:
   Book();                                                              √
   Book(string titleP, string authorP, char[4] publishedP);             √
   string get_title ( ) const;                     //for all accesors √
   string get_author ( ) const;
   void get_published (char year[])const;
   void display_Info( ) const; //display title, author, ISBN, and published
                                                                     √
private:
   string title;                           //for all member variables √
   string author;
   char published[4];
}
```

Question 5.2 **(6)**

```
class Fiction: public Book                                              √
{
public:
   Fiction();                                                           √
   Fiction(string titleP, string authorP, string genreP,
           char[4] publishedP);                                         √
   string get_genre ( ) const;                                          √
   void display_Info( ) const; //display title, author, genre, published   √
private:
   string genre; //e.g. short story, poetry, novel, etc                 √
}
```

Subtract 1 mark for additional data members

Question 5.3

`Display_Info()` is redefined since it has the same signature as in the base class, i.e. the same number of and types of parameters in the derived class as in the base class. ✓
redefining – The definition of a redefined function has the same number and types of parameters in the derived class as the definition in the base class. ✓
overloading – The derived class contains both the original function definition as in the base class as well as the overloaded function. The overloaded function has a different number of parameters or parameters of different types from the function in the base class. ✓ (3)

.

QUESTION 6 [13]

6.1 (5)

```
template <class TKey, class TValue> ✓
class Dictionary
{public:
    Dictionary();
    void Add(TKey key, const TValue &value); ✓
    TValue Find ( TKey key) const; ✓
private:
  vector<TKey>  Keys; ✓
  vector<TValue>      Values; ✓
};
```

6.2 (6)

```
template<class TKey, class TValue>✓
Tvalue✓ Dictionary<TKey, TValue>✓::Find(TKey key) ✓
{
     for(int i = 0; i < Keys.size(); i++)✓
     {
   if(Keys[i] == key) ✓

                return Values[i];
     }
}
```

6.3 (2)

```
Dictionary <string ✓,double ✓> d();
```