

**COS1512
RCO1512**

October/November 2014

INTRODUCTION TO PROGRAMMING II

Duration 2 Hours

75 Marks

EXAMINERS

FIRST
SECOND

MR ES MTSWENI
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**[5]**

- 1 1 Write one C++ statement to accomplish each of the following
- 1 1.1 Declare a vector `v` with base type `int` and initialize the first ten elements in the vector to 0 (1)
- 1 1.2 Add an eleventh element to the vector `v` with the value 15. (1)
- 1 1.3 Change the size of the vector `v` to 40 elements (1)
- 1 1.4 Display both the size and the capacity of the `v` vector (1)
- 1 2 Explain the difference between the size and the capacity of a vector (1)

QUESTION 2**[5]**

Consider the following recursive function

```
int mystery (int number){  
    if (number == 0)  
        return number,  
    else  
        return (number + mystery (number - 1));  
}
```

- 2 1 Identify the base case (1)
- 2 2 Identify the general case (1)
- 2 2 Identify whether the following calls to `mystery()` are valid. If so, give the value returned otherwise explain why the call is invalid (3)
- 2 3.1 `mystery(0)`
- 2 3.2 `mystery(3)`
- 2.3.3 `mystery(-3)`

[TURN OVER]

QUESTION 3**[8]**

The program below converts students' student numbers into their mylife e-mail addresses. It reads one student number at a time from a file called `Students.txt`, converts the student number to the corresponding mylife e-mail address, and then writes it to a file called `StudentsEmail.txt`. File `Students.txt` contains one student number per line, and in file `StudentsEmail.txt` the e-mail addresses should also appear one per line. Each e-mail address is also displayed on the screen. Complete the program by writing down the missing statements.

```
#include <iostream>
```

```
_____ 3.1 _____ //1.Include files needed
```

```
#include <cstdlib>
using namespace std;
int main()
{
```

```
    string stdtnr,
```

```
    _____ 3.2 _____ //2.Declare input file
```

```
    _____ 3.3 _____ //3.Declare output file
```

```
    _____ 3.4 _____ //4.Open input file and check that
                           //file exists
```

```
    _____ 3.5 _____ //5.Open output file and check
                           // that file exists
```

```
    while ( _____ 3.6 _____ ) //6.Extract student number from
                           // input file
```

```
    {
        stdtnr = stdtnr + "@mylife.unisa.ac.za", //create e mail
                                                //address
```

```
        _____ 3.7 _____ //7.write e-mail address to output
                           //file
```

```
        cout << stdtnr << endl,
```

```
    }
```

```
    _____ 3.8 _____ //8.close files
```

```
    return 0;
}
```

[TURN OVER]

QUESTION 4**[9]**

- 4 1 Consider the following code fragment which is assumed to be embedded in a complete and correct C++ program

```
1    int x,  
2    int y,  
3    int *r = &x;  
4    int *s = &y,  
5    *r = 35;  
6    *s = 98,  
7    *r = *s,  
8    cout<<x <<"    "<<y<<endl;  
9    cout<<*r <<"    "<<*s<<endl,
```

- 4 1 1 Give the output of the above code fragment (2)
- 4 1 2 Explain the significance of operator & in line 3. (1)
- 4 1 3 Explain the significance of operator * in line 5 (1)

- 4 2 Consider the following code which is assumed to be embedded in a complete and correct C++ program

```
1  int array_size = 10,  
2  int * a,  
3  a = new int [array_size],  
4  for (int i =0, i < array_size, i++)  
5      a[i] = i,  
6  int * p = a,  
7  cout << p[2],
```

- 4 2 1 Describe the action of the new operator in this code segment. What does the new operator return? (2)
- 4 2.2 Give the output of the above program fragment (1)
- 4 2.3 What is the purpose of the statement in line 6? (1)
- 4 2.4 Write code to return the memory allocated to a, back to freestore (1)

[TURN OVER]

QUESTION 5**[28]**

Use separate compilation to define a class called `Player` that represents a Player in a video game

The class `Player` should contain the following data members

```
string name,           // the name of the player
int level,             // the level to which the player has advanced to
                        // in the game.
int points;           // the number of points accumulated
```

The class should contain the following member functions

- A **default constructor** that initializes `name` to "*Player 0*", `level` to 0 and `points` to 0
- An **overloaded constructor** that accepts three parameters to set the `name`, `level` and `points` to specified values
- A **destructor** that outputs "*Game Over*"
- **Accessor** functions `get_name()`, `get_level()`, `get_points()` that return the values stored in an object's `name`, `level` and `points` member variables respectively
- A void member function called `reset()` that resets the member variables of a `Player` to values specified by parameters
- Overload the **prefix increment operator++** (implemented as a **friend** function) which returns the current instance of the class `Player` after incrementing the `points` by 1. For every 100 points scored, the `level` is also incremented by 1. Use the following prototype:

```
Player operator++(Player& P);
```

 Hint: Use the `%` (modulus) operator
- An **overloaded equality operator >** (implemented as a **friend** function). Use the following prototype

```
bool operator>(const Player& player1, const Player& player2),
```

 This function returns true if `player1`'s `points` are greater than `player2`'s `points` and false otherwise
- An **overloaded operator <<** (implemented as a **friend** function) that displays a `Player`'s `name`, `level` and `points` member variables. Use the following prototype

```
ostream& operator<< (ostream& outs, const Player& P),
```

Attempt the solutions as follows

- 5.1 Create the specification of the class `Player` in the header file `Player.h` (8)

[TURN OVER]

5.2 Create the implementation of the class `Player` including all the `friend` functions (16)

5.3 Fill in the blanks in the following program (4)

```

int main()
{
1.     Player Player1("Jane",1,99),
2.     Player Player2,
3.     __5.3.1__;           //output Player1 and Player2
4.     __5.3.2__;           //increment Player1
5.     if(__5.3.3__)        //test if Player1 beats Player2
6.         cout<< "Player1 wins",
7.         else
8.         cout<<"Player2 wins",
9.         __5.3.4__;       //reset Player2 values to default values
10.    return 0,
}

```

Do not rewrite the program, cite only the line number and give your answer

QUESTION 6

[9]

Study the class `Safe`, which protects an integer value with a password, and answer the questions below

```

class Safe
{
public:
    Safe(int the_item, const string& the_password),
    int open (const string& the_password) const;
private:
    int item;
    string password;
};

```

6.1 Write a template version of the `Safe` class interface so that it may be used to create a `Safe` of any type of item. For example, a `Safe` that contains an item of type `char` or `float`. Do not provide an implementation. Provide only the interface (4)

6.2 Implement the highlighted member function `open()` of the template class `Safe`. The `open()` member function will return the `item` if the passwords match. Use the `assert()` macro to abort the program if the passwords do not match. (4)

6.3 Provide a declaration for a `Safe` as defined in 6.1, containing an item of type of `char`. (1)

[TURN OVER]

QUESTION 7**[11]**

For question 7.1 and 7.2 provide only the interface (i.e. the header files)

7.1 Define a base class named `Rectangle` that contains two data members, `length` and `width`. Class `Rectangle` has two member functions:

- a constructor to create a `Rectangle` with initial values for `length` and `width`, and
- a member function `area` to calculate the area of a `Rectangle`. (2)

7.2 Now derive a class `Box` from class `Rectangle`. The derived class `Box` has an additional data member named `depth`. Class `Box` needs the following member functions:

- a constructor to create a `Box` with initial values for `length`, `width`, and `depth`
- a member function `area` to calculate the surface area of a `Box` (taking the sides of the `Box` into account), and
- a member function `volume` to calculate the volume of a `Box` (4)

7.3 Provide the implementation for the constructor to create a `Box` (2)

7.4 Complete the program below with the necessary calls to the member functions of the objects `R` and `B` (3)

```
#include <iostream>
#include "Rect.h"
#include "Box.h"

using namespace std;

int main()
{
    Rectangle R (4, 5),
    Box B (4,3,2),

    cout << "Area of rectangle R " << 7.4.1 << endl,
    cout << "Total area of box B " << 7.4.2 << endl,
    cout << "Area of bottom of box B " << 7.4.3 << endl;
    return 0;
}
```