

**COS1512
RCO1512**

May/June 2014

INTRODUCTION TO PROGRAMMING II

Duration 2 Hours

75 Marks

EXAMINERS
FIRST
SECOND

MRS P LE ROUX
PROF ID SANDERS

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 8 pages and 9 questions.
Please make sure that you have all 8 pages with the 9 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**[8]**

- 1.1 A photo book store, *Memories*, creates photo books. They requested a program that will manage their billing process. The billing per photo book depends on different criteria. Create three overloaded `computeBill` functions for the photo book store as follows

(2 marks each)

- `computeBill` receives a single parameter which is the price of one photo book ordered. Add 14% tax and return the total due.
- `computeBill` receives two parameters which are the price of a photo book and the quantity ordered. Multiply the two values, add 14% tax, and return the total due.
- `computeBill` receives three parameters. The parameters are the price of a photo book, the quantity ordered, and a coupon value. Multiply the price and quantity, reduce the result by the coupon value, and then add 14% tax, and return the total due.

- 1.2 Write a program named `testBilling` that tests all three overloaded methods (2)

QUESTION 2**[6]**The class `Rectangle`

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

class Rectangle
{
    public
        Rectangle() {}
        Rectangle (int x, int y)    width(x), height(y) {}
        int area() {return width * height;}

        declare friend function duplicate

    private:
        int width, height;
},

int main ()
{
    Rectangle foo,
    Rectangle bar (2,3);

    call to duplicate

    cout << foo.area() << '\n',
    return 0;
}
```

[TURN OVER]

The function `duplicate`, which takes a `Rectangle` instance as input will duplicate the instance

- 2.1 Write the function `duplicate` as a friend function of the class `Rectangle` (3)
- 2.2 Complete the above program by filling in the declaration of the friend function `duplicate` and the call statement to `duplicate` using the variables `bar` and `foo` (3)

QUESTION 3**[5]**

Given the following vector declaration

```
vector<string> SS;
```

write C++ statements to accomplish the following

- 3.1 Add the following string to the vector `SS` (1)
 `"The number is 10"`
- 3.2 Display the size of the vector `SS` (1)
- 3.3 Display the third element of the vector `SS`. (1)
- 3.4 Display the content of the vector `SS` using a `for-loop` (2)

QUESTION 4**[2]**

Consider the following recursive function

```
void tobin(int n)
{
    int m,
    if (n == 0)
    {
        cout << "We are busy with recursion",
    }
    else
    {
        m = n/2;
        tobin(m);
        cout << (n-2 * m) << " ";
    }
}
```

- 4.1 Identify the base case (1)
- 4.2 Give the output produced by the function `tobin` when called as follows (1)
 `tobin(7),`

[TURN OVER]

QUESTION 5**[5]**

The program below opens a file named by the user, reads the contents, computes the average, and reports the average to the user

```
#include <iostream>
#include <cstdlib>

_____1_____ //1.Include files needed
using namespace std;

int main()
{
    char name[13];
    double value = 0, average,
    double sum = 0,
    int count = 0;
    cout << "Enter a file name " << endl,
    cin >> name,

    _____2_____ //2.Declare input file

    _____3_____ //3.Open the input file and check that the
    // file exists

    input >> value,
    while (_____4_____) //4.Extract an existing session from the
    // input file

    {
        count=count+1;
        sum = sum + value;
        input >> value;
    }
    average = sum / count;
    cout << "Average of " << count << " numbers is "
    << average << endl,

    _____5_____ //5.close files

    return 0;
}
```

QUESTION 6**[10]**

6 1 What does the following declaration do?

(1 mark each)

```
6 1 1 int * pOne,
6 1 2 int vTwo,
6 1 3 int * pThree = &vTwo,
```

[TURN OVER]

6 2.1 Explain the difference between the following two uses of the operator *? (2)

```
int * q = p,  
n = *p,
```

6 2.2 Explain the difference between the following two uses of the reference operator & (2)

```
int &r = n,  
p = &n,
```

6 3 What is the output of the following code fragments? (1 mark each)

6 3 1

```
int arraySize = 10,  
int * anArray,  
anArray = new int [arraySize],  
  
for (int i = 0, i < arraySize, i++)  
    anArray[i] = i,  
while (*anArray < 9)  
{  
    anArray++,  
    cout << *anArray << " ",  
}  
cout << endl;
```

6 3 2

```
int arraySize = 10,  
int * anArray;  
anArray = new int [arraySize],  
int *p = anArray,  
for (int i = 0, i < arraySize, i++)  
    anArray[i] = i,  
p[0] = 10,  
for (int i = 0; i < arraySize, i++)  
{  
    cout << anArray[i] << " ",  
}  
cout << endl,
```

6 3 3

```
int arraySize = 10,  
int * anArray,  
anArray = new int [arraySize],  
for (int i = 0, i < arraySize, i++)  
    *(anArray + i) = i,  
  
for (int i = 0; i < arraySize, i++)  
    cout << anArray[i] << " ",  
cout << endl;
```

QUESTION 7**[23]**

A class called `Product` represents a single product sold by an office supply store. It contains the following data members:

```
long id,           //5 digit product ID number
double price,      //wholesale price
double markup,     //markup from wholesale as a percentage
int number;        //number of products in stock
```

The class should contain

- a default constructor
- a constructor that initializes all data members
- a function called `display()` that displays the details of a product on the screen
- a function called `retailPrice()` that returns a product's price increased by the markup percentage
- a function `modify()` to allow the markup percentage to be modified
- a function `increment()` which returns the current instance of the class `Product` after incrementing `number` by 1
- a function `decrement()` which returns the current instance of the class `Product` after decrementing `number` by 1

- 7.1 Create the header file `Product.h` that contains the `Product` class specification (8)
- 7.2 Create the implementation of the class `Product` (10)
- 7.3 What will be the output produced by the code below? (2)

```
int main()
{
    Product P1(1111,10.00,20,100),
    P1.increment();
    P1.decrement(),
    P1.decrement();
    P1.display(),
    cout << "Retail Price    :" << P1.retailPrice();
    return 0;
}
```

- 7.4 The `++` operator in C++ is called the increment operator. If `n` is a variable of `int`, the `n++` increases the value of `n` by 1. Overload the increment operator to perform the same function as the `increment()` function (3)

[TURN OVER]

QUESTION 8**[8]**

Study the class `ComplexNumber` and answer the questions below

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

class ComplexNumber
{
    public.
        ComplexNumber(),
        ComplexNumber(int realPart),
        ComplexNumber(int r, int i);
        void output(char& i),
        ComplexNumber add(ComplexNumber x);

    private
        int real,
        int imagine;
},
```

- 8 1 Write a template version of the `ComplexNumber` class interface so that it may be used to create a `ComplexNumber` of any type, e.g. a `ComplexNumber` of `ints`. Do not provide an implementation. Provide only the interface (4)
- 8 2 Implement the highlighted constructor - `ComplexNumber(int r, int i)` - of the template class `ComplexNumber` (3)
- 8 3 Provide a declaration for a `ComplexNumber` of `double` (1)

QUESTION 9**[8]**

For question 9 1 and 9 2 provide only the interface

- 9 1 We want to design a class for a `Door`. Every `Door` is in one of two states: open or shut. Use the enumeration type `enum state {open, shut};` to define the only data member `door_status`. When a new `Door` is created, it will initially be shut. A `Door` needs the following methods (3)
- A constructor to create the `Door`
 - `isOpen` which returns 1 if the door is open and 0 otherwise
 - `open` which opens the door
 - `close` which closes the door
- 9 2 Now define a class for a `LockableDoor`. This class must be derived from the `Door` base class. A `LockableDoor` has greater functionality than a plain door. It knows whether or not it is locked, it knows how to lock and unlock itself. Lockable doors are also shut or open. A lockable door cannot open itself unless it is unlocked. It needs the following methods (3)

[TURN OVER]

- A constructor
- `isLocked` which returns 1 if the door is locked and 0 otherwise
- `open` to open the door
- `lock` to lock the door
- `unlock` to unlock the door

9.3 Create an instance of a `Door` and write C++ code to change its status- from close to open or open to close (2)