

## **COS1512**

May/June 2012

### **INTRODUCTION TO PROGRAMMING 2**

Duration 2 Hours

75 Marks

EXAMINATION PANEL AS APPOINTED BY THE DEPARTMENT.

---

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

**This examination paper remains the property of the University of South Africa  
and may not be removed from the examination room.**

#### **Instructions / Instruksies:**

- 1 Answer all questions
- 2 All rough work must be done in your answer book
- 3 The mark for each question is given in brackets next to the question
- 4 Please answer the questions in order If you want to do a question later, leave enough space

**GOOD LUCK!**

**QUESTION 1****[5]**

- 1 1 Explain what is wrong with the following code

```
char funny[] = "uhuhuh";
for (int k = 0; k < 7; k++ )
    if (funny[k] == 'h')
        funny[k] = 'u';
```

(1)

- 1 2 Consider the following C string definition

```
char name[7];
```

Why is the following assignment statement not valid?

```
name = "Johnny";
```

Give an example of a valid assignment to name .

(2)

- 1 3 What output would be produced from the following C++ code fragment

(2)

```
vector<char> myName;
myName.push_back('E');
myName.push_back('l');
myName.push_back('l');
myName.push_back('z');
myName.push_back('e');
for(int i = 1; i < 5; i+=2)
{
    myName[i] = '(';
}
myName.push_back(')');
for (int i = 0; i < myName.size(); i++)
{
    cout<< myName[i];
}
```

**QUESTION 2****[5]**

- 2 1 Consider the following incomplete code fragment and answer the questions that follow.

```
1.   int x = 10;
2.   int y = 12;
3.   int *p1 = &y;
4.
5.
6.   int *p3 = p1;
7.   delete p1;
8.   cout << *p3 << " " << *p2 ;
```

- 2 1 1 For line 4, give a C++ statement to declare an int pointer p2 and let it point to variable x

(1)

- 2 1 2 For line 5, give a C++ statement to display the value pointed to by p2 without referring to the variables x or y.

(1)

- 2.1.3 What will be displayed by line 8, taking into consideration the code that you have added for lines 4 and 5? (1)
- 2.1.4 What is the function of the statement in line 7? (1)
- 2.2 What is a static array? (1)

**QUESTION 3****[35]**

The main program in this question is used to manage a user's shareholdings. The program inputs a new purchase of shares, checks a file of shares `Portfolio.dat` to see whether the user already owns shares in the same company, and if so, updates the shareholding for that company. This produces an updated version of the file of shares `NewPortfolio.dat`. At the same time, the total value of the shares is determined.

Using the input file `Portfolio.dat` below:

**Input file `Portfolio.dat`:**

```
OldMutual 10 12.5
Sanlam 15 10.5
Allied 5 6.5
```

the program produces the following output

**Output:**

```
Enter name of company in which shares are bought: OldMutual
Enter number of shares bought: 10
Enter unit value of one share: 20.5
```

```
Total value of shares: R 600.00
```

```
Press any key to continue . . .
```

**Output file `NewPortfolio.dat`:**

```
OldMutual 20 20.5
Sanlam 15 10.5
Allied 5 6.5
```

To implement this program, define a class `Shares` that represents shares for a shareholder in a specific company on the JSE. This class has three member variables:

- `company`, a string that holds the name of the company
- `nrShares`, an integer value that indicates the number of shares this shareholder owns and
- `unitValue`, a float value that indicates the current value of one share

In addition, the class should have the following member functions.

- A **default constructor** that initializes `company` to an empty string. `nrShares` and `unitValue` should each be initialized to 0
- An **overloaded constructor** that represents shares bought in a new company and sets `company`, `nrShares` and `unitValue` to specified values.
- A **destructor** that does not perform any action
- Accessor functions for the member variables.
- An **overloaded equality operator** `==` to compare two `Shares` objects. The `==` operator is implemented as a **friend** function with the following prototype  

```
bool operator==(const Shares & s1, const Shares & s2)
```

This function returns `true` if `shares1` and `shares2` have the same company name, and `false` if not

- An overloaded **operator+** for class Shares so that the following are feasible in the main program **a = b + c**, where **a, b**, and **c** are all Shares objects. The overloaded **operator+** should add the values of the **nrShares** member variables. The **company** and **unitValue** member variables should be set to the **company** and **unitValue** member variables of the second operator. Use the following prototype  
`Shares operator+ (const Shares & s1, const Shares & s2);`
- An overloaded extraction **operator >>** (implemented as a **friend** function) so that it can be used to input values of type Shares
- An overloaded insertion **operator <<** (implemented as a **friend** function) that outputs all the member variables of a Shares object

3.1 Create the header file `Shares.h` that contains the Shares class specification (8)

3.2 Create the implementation of the class Shares including all the friend functions. (16)

3.3 Complete the application program (`main()`) below by citing the number and writing down the missing statement. This program obtains the detail for a new purchase of shares from the user and creates an object to represent the new purchase of shares. It then extracts all existing shares in the user's portfolio one by one from a file `Portfolio.dat` and checks whether the shareholder already has shares in the company. If the user already holds shares in the company, the **operator+** is used to update the shares in the company to include the new shares purchased. All shares (updated as well as unchanged existing shares) are output to a new portfolio file called `NewPortfolio.dat`. If the user does not already own shares in the company, the new purchase of shares are simply output to `NewPortfolio.dat`. Before outputting a shareholding to `NewPortfolio.dat`, the program calculates the value of the shareholding (number of shares \* unit value per share) and adds it to a running total. The running total is used to calculate the total value of the shares, which is displayed on the console window at the end of the program. Be sure to use appropriate member functions in the required statements (11)

```
#include <iostream>
#include <iomanip>

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

int main()
{
    _____ 2 _____ //2.Declare input and output files
    _____ 3 _____ //3.Open the file Portfolio.dat and check
                        // that the file exists
    _____ 4 _____ //4.Open the file NewPortfolio.dat and
                        // check that the file exists

    //Add a new purchase of shares
    string companyBought;
    int nrBought;
    float shareValue;

    cout << "Enter name of company in which shares are bought: ";
    cin >> companyBought;
    cout << "Enter number of shares bought: ";
    cin >> nrBought;
    cout << "Enter unit value of one share: ";
```

```

cin >> shareValue;

    5 //5.Instantiate object newShare for the
    // new share bought
Shares aShare; //instantiate object aShare to extract an
//existing share from file Portfolio.dat
float totalSharesValue = 0;

while ( 6 ) //6.Extract an existing share from file
    // Portfolio.dat
{
    if ( 7 ) //7.Compare newShare with aShare to
    //check whether shareholder already
    //owns shares in the company
    {
        8 //8.Update shares to include newly
        //bought shares
    }

    cout.setf(ios::showpoint);
    cout.setf(ios::fixed);
    cout << setprecision(2);

    totalSharesValue += 9 //9. Calculate and add
    // current value of shares for this company
    10 //10.Output share to file NewPortfolio.dat
}

cout << "\nTotal value of shares: R " << totalSharesValue << endl
    << endl;
11 //11.close files
return 0;
}

```

**QUESTION 4****[15]**

Consider the following class:

```

class InsurancePolicy
{
public:
    InsurancePolicy();
    InsurancePolicy(int pNr, string pHolder, double aRate);
    void showPolicy(ostream & out)const,
    void setPolicy(int pNr, string pHolder, double aRate);
    int get_pNr()const;
    string get_pHolder()const;
    double get_aRate()const;
private:
    int policyNr;
    string policyHolder;
    double annualRate;
};

```

41 Derive a class CarInsurance from class InsurancePolicy. This class has an additional member

variable, excess. Class InsurancePolicy also has member functions, get\_excess() and set\_excess() to return member variable excess and update member variable excess respectively. The class CarInsurance should override function showPolicy() in order to display the member variables of CarInsurance and function setPolicy() in order to update the member variables of CarInsurance. Provide only the interface of class CarInsurance in terms of a header file. The header file should contain compiler directives to prevent multiple definitions. Assume that the interface of class InsurancePolicy is contained in an interface file called InsurancePolicy.h (8)

4 2 Implement the overloaded constructor for the class CarInsurance by invoking the base class constructor (3)

4 3 Consider the following implementation

```
void CarInsurance:: setPolicy(int pNr, string pHolder,
                             double aRate, double eValue)
{
    policyNr = pNr;
    policyholder = pHolder;
    annualRate = aRate;
    excess = eValue;
}
```

Explain why setPolicy() is not a legal definition in the derived class CarInsurance? How should this problem be resolved? (2)

4 4 From your implementation of class CarInsurance give an example of

4 4 1 overloading a function, and (1)

4 4 2 redefining a function (1)

## QUESTION 5

[10]

Consider the following class:

```
class PQueue {
public:
    PQueue() {}
    int size() const;
    bool isEmpty() const;
    void enqueue(int elem);
    int extractMax();
private:
    vector<int> list;
};
```

The class PQueue has the following operations:

size returns the size of the list  
isEmpty return true if the list is empty.  
enqueue inserts an element at the end of the list  
extractMax returns and removes the largest element from the list

5 1 Write a template version of the PQueue class. In other words, redesign the PQueue interface so that it may be used to create a PQueue of any type. For example, a PQueue of chars, or a PQueue of strings (Do not provide any implementations). Provide only the interface (6)

5 2 Implement the enqueue member function of the template class PQueue (3)

5 3 Provide a declaration for a PQueue of strings (1)

**QUESTION 6** [5]

The program below contains an incomplete recursive function `raised_to_power()`. The function returns the value of the first parameter number of type float raised to the value of the second parameter power of type int for all values of power greater than or equal to 0

```

1. #include <iostream>
2. using namespace std;
3. float raised_to_power(_____)
4. {
5.     if (power < 0)
6.     {
7.         cout << "\nError - can't raise to a negative power\n";
8.         exit(1);
9.     }
10.    else if (_____)
11.        return (_____),
12.    else
13.        return (number * raised_to_power(number, power - 1));
14. }
15. main()
16.
17. float answer = raised_to_power(4.0,3);
18. cout << answer;
19. return 0;
20.}

```

6 1 Complete the function header in line 3 (1)

6 2 Using the fact that any value raised to the power of 0 is 1, complete the base case in line 10 and 11 (2)

6 3 Why do we need a base case in a recursive function? (1)

6 4 What is the purpose of the general case? (1)

**EXAMINATION PANEL AS APPROVED BY THE DEPARTMENT:**

**First Examiner: Mrs MA Schoeman**

**Ms CNA Nombewu**

**Second Examiner: Prof I Sanders**