

**COS1512
COS112V**

May/June 2011

INTRODUCTION TO PROGRAMMING 2

Duration 2 Hours

75 Marks

EXAMINERS
FIRST
SECOND

MS MA SCHOEMAN
MS P LE ROUX

MS HW DU PLESSIS

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 sms[] = "Gr8",
strcat(sms, " & ");
```

NB

strcat concatenates " & ." to the end of sms

(1)

- 1 2 Explain the difference between

```
char short_string[] = "abc",
char short_string[] = {'a', 'b', 'c'}
```

(2)

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

(2)

```
vector<int> myList(4),
myList[0] = 2;
for(int i = 1; i <= 3, i++)
{
    myList[i] = 2*myList[i - 1];
}
myList.push_back(34),
for (int i = 0, i < myList.size(), i++)
{
    cout<< myList[i]<<" ";
}
```

QUESTION 2**[5]**

- 2 1 Consider the following code segment and answer the questions that follow

```
int x = 10,
int y = 12,
int *px = &x,
int *py = &y;
int *p = new int;
int temp,
```

- 2 1 1 Write a short code segment to swap the values pointed to by px and py without referring to variables x

and y

(2)

- 2 1 2 Give a C++ statement to release the memory occupied by p

(1)

- 2 1 3 Explain what the following line of code would imply?

px = py,

(1)

- 2 2 Explain the difference between static arrays and dynamic arrays

(1)

QUESTION 3

[35]

Members of a certain medical aid fund receive free gym membership, on the provision that they visit a gym a specified number of times per month. Once a year, members who do not comply with this provision, receive a message to this effect, before their gym membership are terminated. Define a class `GymMember` that can be used to represent members of this medical aid. The class `GymMember` has three member variables:

- `name`, a string that holds the name of the gym member
- `cellNo`, a string that holds the gym member's cell number
- `nrVisits`, an integer containing the number of times the gym member has visited the gym during the past year

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

- A **default constructor** that initializes `name` and `cellNo` respectively to an empty string, and `nrVisits` to 0
- An **overloaded constructor** that creates a new gym member and sets `name`, `cellNo` and `nrVisits` to specified values
- A **destructor** that does not perform any action
- **Accessor** functions `get_name()`, `get_cellNo()` and `get_nrVisits()` to return the values stored in an object's `name`, `cellNo` and `nrVisits` member variables respectively
- A **Mutator** function `set_cellNo()` to update a gym member's cell phone number
- Overload the **prefix increment operator** `++` (implemented as a **friend** function) to return the current instance of the class `GymMember` after incrementing the `nrVisits` by 1. Use the following prototype:
`GymMember operator++(GymMember & m);`
- An **overloaded extraction operator** `>>` (implemented as a **friend** function) so that it can be used to input values of type `GymMember`
- An **overloaded insertion operator** `<<` (implemented as a **friend** function) that displays a gym member's name, cell phone number and the number of visits to the gym during the past year

You should attempt the solutions as follows:

- 3.1 Create the header file `GymMember.h` that contains the `GymMember` class specification (10)
- 3.2 Create the implementation of the class `GymMember` including all the friend functions (13)
- 3.3 Complete the application program (`main()`) below by citing the number and writing down the missing statement. This program obtains the name of a gym member from the user and searches for the gym member in a file called `GymMembers.dat`. If the gym member is found, the user can update either the number of visits or the cell number for the user. If the gym member is not found, the user has the option to add this person as a new gym member. Be sure to use appropriate member functions in the required statements (12)

```
#include <iostream>
    1
using namespace std;
```

// 1. Include file(s) required

```
int main()
{
    string memberName;
    cout << "Enter the member's name: ",
    cin >> memberName;
```

2

// 2. Declare input file

3

// 3. Open the file `GymMembers.dat` and
//test that file exists

```
GymMember aMember;
bool found = false;
```

```
while (!(found) && (____4____))    //4. Read in a gym member into object
                                   //aMember from file
{
    if (____5____)                //5. Check if aMember's name is the
                                   //same as memberName
    {
        found = true;
    }
}

if (found)
{
    char update = ' ';
    //determine type of update on object
    while (update != 'v' && update != 'c')
    {
        cout << "Update visits(v) or cell number(c)?";
        cin >> update;
    }

    if (update == 'v')    //number of visits should be updated
    {
        _____6_____    //6. Update number of visits

        // 7. Calculate average number of visits per month for this member and display
        cout << memberName << " has an average of " << _____7_____
              << " visits per month for this year" << endl;
    }
    else    // cell number should be updated
    {
        string newCell;
        cout << "Enter new cell number: ";
        cin >> newCell;
        _____8_____    //8. Update cell number

        //9 - 10. Display the name and cell number of object aMember
        cout << "New cell number for " << _____9_____ << " is "
              << _____10_____ << endl,
    }
}

else    //membername is not a member of the gym
{
    char answer,
    string cell,
    cout << memberName << " is not a member of this gym. "
        << "Add as new member? (y/n)" << endl;
    cin >> answer,
    if (answer == 'y')
    {
        cout << " Enter cell number for " << memberName << ". ",
        cin >> cell;
        _____11_____    //11. Instantiate a new object for the new member
    }
}
```

```

    }
}
    12
    return 0;
}
//12. Close input file

```

QUESTION 4**[15]**

Consider the following class

```

class Computer
{
public:
    Computer();
    Computer(int s, int mm, int hm);
    void display_specs(ostream & out) const;
    int get_speed() const;
    int get_mmemory() const;
    int get_hmemory() const;
private:
    int speed;
    int main_memory;
    int harddisk_memory;
};

```

4.1 Derive a class Laptop from class Computer. This class has additional member variables, battery_time and weight. Class Laptop also has member functions, get_battime() and get_weight() to return member variables battery_time and weight respectively. The class Laptop should override function display_specs() in order to display the member variables of Laptop. Provide only the interface of class Laptop in terms of a header file. The header file should contain compiler directives to prevent multiple definitions. Assume that the interface of class Computer is contained in an interface file called "Computer.h" (7)

4.2 Implement the overloaded constructor for the class Laptop by invoking the base class constructor (4)

4.3 Consider the following implementation

```

void Laptop::display_specs(ostream & out) const
{
    out << "Speed : " << speed << endl;
    out << "Main memory : " << main_memory << endl;
    out << "harddisk_memory : " << harddisk_memory << endl;
    out << "Battery_time : " << battery_time << endl;
    out << "Weight " << weight << endl;
}

```

Explain why display_specs() is not a legal definition in the derived class Laptop? Suggest TWO possible ways of resolving this problem (3)

4.4 Is function display_specs() an example of overloading? Explain your answer (1)

QUESTION 5**[10]**

Consider the following class

```
class Storage
{
public:
    Storage(int size);
    int getItemAtPosition(int pos),
    void storeItem(int pos, int item);
    bool find (int item);
private:
    vector<int> compartments;
};
```

The class Storage has the following operations

getItemAtPosition	– returns the element at position, pos from compartments
storeItem	– stores element, item at position pos in compartments
find	– returns true if element item was found in compartments

- 5.1 Write a template version of the Storage class. In other words, redesign the Storage interface so that it may be used to create storage of any type of item. For example, Storage that contains objects of type Car or Product. Provide only the interface. (5)
- 5.2 Provide the implementation of function storeItem from the template class Storage. (4)
- 5.3 Provide a declaration for a Storage object intended to contain 10 objects of type of Car. (1)

QUESTION 6**[5]**

Function iterativeSum() below returns the sum of the first n elements of an array. So for example, assuming the following declaration

```
int arr[6] = {1,2,3,4,5,6};
```

the call iterativeSum (arr,6) will return 21

```
int iterativeSum( int arr[], int n )
{
    if ( n == 0 )
        return 0;
    else{
        int sum = 0;
        for(int i = 0; i < n, i++){
            sum = sum + arr[i];
        }
        return sum;
    }
}
```

To rewrite function iterativeSum() as a recursive function recursiveSum(), we need to specify a base case and a general case

- 6.1 What is the purpose of the base case? (1)

- 6.2 What is the purpose of the general case? (1)
- 6.3 Use the code below for the general case for function `recursiveSum()`, and write down the complete function `recursiveSum()` (3)
- ```
return arr[size - 1] + recursiveSum(arr, size-1);
```