

COS1512

October/November 2013

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 7 pages

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!

[TURN OVER]

QUESTION 1**[5]**

Consider the following C++ code fragment

```
1 char name[],
2 name[] = "James Khumalo";
3 for(int i = 0; name[i] != '\0', i++){
4 if(name[i] == 'm'){
5 name[i] = '*';
6 }
7 }
8 cout<<name;
9 cout<<endl,
```

- 1.1 Explain what is wrong with the code in *line 2* (1)
- 1.2 Write a statement to correct the error in line 2 (1)
- 1.3 What distinguishes a C-string variable from an ordinary array? (1)
- 1.4 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]**

For each of the following, write a single statement that performs the indicated task

- 2.1 Declare both variable `ptr1` and `ptr2` to be a pointer to an integer
- 2.2 Create a new dynamic array with ten integer elements to which `ptr1` points
- 2.3 Set pointer `ptr2` so that it points to the same address as `ptr1`
- 2.4 Assign the value 53 to the first element in the dynamic array to which `ptr1` points
- 2.5 Release the memory location to which `ptr1` points to the freestore memory

[TURN OVER]

QUESTION 3**[35]**

A keen runner and programmer keeps meticulous records of his training for Comrades. For each training session he records the date, the distance trained, and the duration of the training session, in a file of objects, `Training.dat`. Define a class `TrainingSession` that represents one such training session. This class has three member variables:

- `date`, a string that holds the date of the session
- `distance`, a double value that indicates the distance covered and
- `time`, a string that indicates the duration of the training session

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

- A **default constructor** that initializes `date` and `time` each to an empty string. `distance` should be initialized to 0.
- An **overloaded constructor** to set `date`, `distance` and `time` to specified values.
- A **destructor** that does not perform any action.
- Accessor functions for the member variables.
- An **overloaded operator>** to compare two `TrainingSession` objects. The `operator>` is implemented as a **friend** function with the following prototype:

```
bool operator>(const TrainingSession& t1, const TrainingSession& t2)
```

This function returns `true` if `distance` for `t1` has a larger value than `distance` for `t2`, and `false` if not.
- An overloaded extraction **operator >>** (implemented as a **friend** function) so that it can be used to input values of type `TrainingSession`.
- An overloaded insertion **operator <<** (implemented as a **friend** function) that outputs all the member variables of a `TrainingSession` object.

You should attempt the solutions as follows:

- 3.1 Create the header file `TrainingSession.h` that contains the `TrainingSession` class specification. (8)
- 3.2 Create the implementation of the class `TrainingSession` 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 details for one training session from the user and instantiates (creates) an object to represent the training session. The program then updates file `Training.dat` by adding this training session to a new file called `UpdatedTraining.dat` and copying all the previous training sessions from file `Training.dat` to `UpdatedTraining.dat`. In the process the program also calculates the total distance trained to date and determines the longest training session. Finally the program displays the details of the longest training session as well as the total distance trained to date, on the console window. Be sure to use appropriate member functions in the required statements.

[TURN OVER]

For example, if the input file has the following contents

Input file (Training.dat)

```
1/1/2013    23 5  2h30min
3/1/2013    12    1h20min
5/1/2013    15    1h59min
6/1/2013    25    3h3min
```

The program should produce the following output

Console output:

```
Enter date of training session  3/3/2013
Enter distance trained: 22.3
Enter duration of training: 2h59min
```

```
Total distance trained to date 97.80 km
```

```
Longest training session date, distance and time:
```

```
6/1/2013 25.00 3h3min
```

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

Output file (UpdatedTraining.dat)

```
3/3/2013    22.3  2h59min
1/1/2013    23.5  2h30min
3/1/2013    12    1h20min
5/1/2013    15    1h59min
6/1/2013    25    3h3min
```

(14)

```
#include <iostream>
_____ 1 _____ //1.Include files needed
#include <iomanip>
using namespace std;

int main()
{
    _____ 2 _____ //2.Declare input and output files

    _____ 3 _____ //3.Open file Training.dat and check that the
                          // file exists

    _____ 4 _____ //4.Open file UpdatedTraining.dat and check that
                          // the file exists

//Enter a new training session
string tDate;
double tDistance,
string tTime;

cout << "Enter date of training session: ";
cin >> tDate;
cout << "Enter distance trained: ",
cin >> tDistance,
```

[TURN OVER]

```

cout << "Enter duration of training: ";
cin >> tTime;

_____5_____ //5.Instantiate object tTrain for the new training
                // session

_____6_____ //6.Insert tTrain in file UpdatedTraining

_____7_____ //7.Instantiate object longestTraining to the default
                // constructor

_____8_____ //8.Compare longestTraining with tTrain and update
                // longestTraining if necessary

//initialise totalDistance with today's training distance
double totalDistance = tDistance;

_____9_____ //9.Instantiate object aTraining to extract a
                // training session from file Training.dat

while (_____10_____) //10.Extract an existing session from
{
    _____11_____ //11.Compare longestTraining with aTraining and update
                        // longestTraining if necessary

    _____12_____ //12.Calculate total distance

    _____13_____ //13.Output training session to file UpdatedTraining.dat
}

cout.setf(10s::showpoint),
cout.setf(10s::fixed),
cout << setprecision(2);
cout << "\nTotal distance trained to date " << totalDistance << " km"
    << endl << endl;
cout << "Longest training session date, distance and time: " << endl,
cout << longestTraining;

_____14_____ //14.close files

return 0,
}

```

QUESTION 4**[15]**

Consider the class specification (interface) for the class BankAccount below

```

class BankAccount
{
public
    BankAccount();
    BankAccount(int accNr, double initBalance),

```

[TURN OVER]

```

void setAccNr(int nr);
int getAccNr( ) const;
double getBalance( ) const;
void deposit(double amnt),
void withdraw(double amnt);
void displayAccInfo( ) const; //display accNr and balance
private
    int accNr;
    double balance,
}

```

- 4.1 Derive a class `SavingsAccount` from class `BankAccount`. This class has an additional member variable `interestRate` representing the interest rate on the savings account. The class also has additional member functions `setInterestRate()`, `getInterestRate()` and `PostInterestRate()`. Member function `postInterestRate()` adds earned interest to the balance. The class should override member function `displayAccInfo()` to display the account number, current balance and interest rate. Provide only the interface of class `SavingsAccount` in terms of a header file. The header file should contain compiler directives to prevent multiple definitions. Assume that the interface of class `BankAccount` is contained in an interface file called `BankAccount.h`. (9)

- 4.2 Implement the overloaded constructor for the class `SavingsAccount` by invoking the base class constructor. (3)

- 4.3 Consider the following code

```

SavingsAccount myAccount,
myAccount.BankAccount: displayAccInfo();

```

Which version of `displayAccInfo()` will be executed with this function call, the one in `BankAccount` or the one in `SavingsAccount`? (1)

- 4.4 Member function `postInterestRate()` adds earned interest to the balance. Show how to adapt class `BankAccount` to allow `postInterestRate()` to access the member variable `balance` by name and explain why the adaptation is done in this way. (2)

QUESTION 5

[10]

Consider the class below and answer the questions that follow

```

class Triplet
{
public:
    Triplet(int x, int y, int z); //default constructor
    vector<int> makecopy() const; //make an exact copy of the vector
    void rotate(); //rotate elements of the vector one position right
    int get()const; //return value in first position of vector

```

[TURN OVER]

```
private:
    vector<int> v;
}
```

- 5 1 Write a template version of the `Triplet` class interface so that it may be used to create a `Triplet` of any type, e.g. a `Triplet` of `chars` or a `Triplet` of `strings`. Do not provide an implementation. Provide only the interface (5)
NB a `Triplet` is a set of three values, these values are stored in a vector
- 5 2. Implement the constructor of the template class `Triplet` (3)
- 5 3. Provide a declaration for a `Triplet` of `strings` (2)

QUESTION 6**[5]**

Consider the following recursive function

```
string b (int n)
{
    string s;
    if (n%2 == 0),
        s = "0"
    else s = "1";
    if (n < 2)
        return s;
    return b(n/2) + s,
}
```

- 6 1 Identify the base case (1)
- 6 2 Identify the general case (1)
- 6 3 Function `b` is called in the following program fragment.

```
string result = b(8),
cout << "8 converted by function b() is " << result;
```
- 6 3 1 What is the output produced by this program fragment? (1)
- 6 3 2 What does function `b()` calculate? (2)