COS1512 ASSIGNMENT 4 2023

TUTORIAL MATTER:	**Chapters 14, 15 and 17 of the Study Guide (Tutorial Letter 102 available under Additional Resources on the COS1512 website on myUnisa)** **Chapters 14 (excluding section 14.3), 15 (excluding sections 15.2 and 15.3) and 17 of Savitch**
WEIGHT:	**25%**
QUIZ AVAILABLE:	**26 August 2023**
DUE DATE:	**18 September 2022**
CUT-OFF DATE	**18 September 2022**

NB: This assignment consists of two parts:

- **a part where you write and implement program code (this part) and**
- **an MCQ part where you answer questions on the code you have written, and the material covered in this assignment.**

The MCQ part of the assignment will be available in the Assessment Shell for Assignment 4 on the myModules site for COS1512.

You will not be able to do the MCQ part unless you have completed the coding part.

Question 1

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.

The algorithm used in this question to write a recursive function to raise a float value `number` to a positive `power` uses repeated multiplication as follows:

$\text{number}^{\text{power}}$	= 1	if power= 0
	= number x $\text{number}^{\text{power-1}}$	otherwise

In other words, `number` raised to `power` gives 1 if `power` is 0;

Open Rubric

and otherwise $\texttt{number}^{\texttt{power}}$ can be calculated with the formula:

$\texttt{number x number}^{\texttt{power-1}}$

```
1. #include <iostream>using namespace std;
2. float raised_to_power(________________)
3. {
4.  if (power < 0)
5.  {
6.     cout << "\nError - can't raise to a negative power\n";
7.                    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.}
```

(a) Complete the function header in line 3.

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

(c) Why do we need a base case in a recursive function?

(d) What is the purpose of the general case?

Question 2

Examine the code fragment below and answer the questions that follow:

```
1: #include <iostream>
2: using namespace std;
3:
4: //-----------------------------------------------------------
5:
6: class A
7: {
8:    private:
9:        int x;
10:  protected:
11:       int getX();
12:  public:
13:       void setX();
14: };
```

```cpp

int A::getX()
{
  return x;
}


void A::setX()
{
     x=10;
}

//--------------------------------------------------------------
class B
{
 private:
     int y;
 protected:
     A objA;
     int getY();
 public:
     void setY();
};

void B::setY()
{
     y=24;
     int a = objA.getX();
}

//--------------------------------------------------------------

class C: public A
{
 protected:
     int z;
 public:
     int getZ();
     void setZ();
};

int C::getZ()
{
     return z;
}

 void C::setZ()
{
     z=65;
}
```

Answer the following questions based on the code fragment given above:

(a) Is line 18 a valid access? Justify your answer.

(b) Is line 32 a valid statement? Justify your answer.

(c) Identify another invalid access statement in the code.

(d) Class C has `public` inheritance with the class A. Identify and list class C's `private`, `protected` and `public` member variables resulting from the inheritance.

(e) If class C had `protected` inheritance with the class A, identify and list class C's `private`, `protected` and `public` members variables resulting from the inheritance.

Question 3

Consider the class definition below and answer the questions that follow:

```
class InsurancePolicy
{

public:
    InsurancePolicy();
    InsurancePolicy(int pNr, string pHolder, double aRate);
    ~InsurancePolicy();
    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;
};
```

(a) Implement the class `InsurancePolicy.`

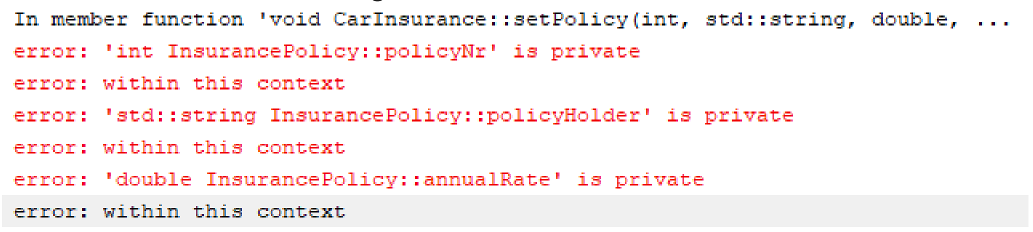
(b) Code the interface for a class `CarInsurance` derived from class `InsurancePolicy` (the base class). This class has an additional member variable, `excess`. Class `InsurancePolicy` also has member functions, `get_excess()` and `set_excess()` to return the value of member variable `excess` and update the value of member variable `excess` respectively. The class `CarInsurance` should override function `showPolicy()` in order to display the member variables of `CarInsurance` and also override member function `setPolicy()` in order to update the member variables of `CarInsurance.`

(c) Implement the class `CarInsurance` and use the code below to implement `setPolicy()`:

```
void CarInsurance:: setPolicy(int pNr, string pHolder,
double aRate, double eValue)
{

policyNr = pNr;
policyholder = pHolder;
annualRate = aRate;
excess = eValue;
}
```

You should obtain the following errors:



```
In member function 'void CarInsurance::setPolicy(int, std::string, double, ...
error: 'int InsurancePolicy::policyNr' is private
error: within this context
error: 'std::string InsurancePolicy::policyHolder' is private
error: within this context
error: 'double InsurancePolicy::annualRate' is private
error: within this context
```

Explain why `setPolicy()` is not a legal definition in the derived class `CarInsurance`?

Suggest two ways to fix this problem.

(d) Add a member function
`void showPolicy(ostream & out)const;`
to the class `InsurancePolicy` as well as to the class `CarInsurance` in order to display the member variables of `InsurancePolicy` and `CarInsurance`.

(e) Use the following driver program to test your classes `InsurancePolicy` and `CarInsurance`:

```
#include <iostream>
#include <fstream>
#include "Insurance.h"
#include "CarInsurance.h" using namespace std;

int main()
{

    InsurancePolicy myPolicy(123456, "Peter Molema", 3450.67);
    CarInsurance yourPolicy(456891, "Wilson Ntemba", 5550.67,
                                            15000.00);
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    myPolicy.showPolicy(cout);
    cout << endl;
    yourPolicy.showPolicy(cout);
    cout << endl << "AFTER UPDATES:" << endl;

    myPolicy.setPolicy(123456, "Peter Molema", 5450.67);
    yourPolicy.setPolicy(456891,"WilsonNtemba",6650.67,
    25000.00);
    myPolicy.showPolicy(cout);
    cout << endl;
    yourPolicy.showPolicy(cout);
    cout << endl;

    return 0;
}
```

Question 4

(a) Write a function called `found()` to determine whether a specific value occurs in a vector of integers. The function should receive two parameters: the vector to be searched (`v`) and the value to search for (`val`). The function `found()` should return a Boolean value to indicate whether or not `val` occurs in vector `v`.

Test your function `found()` in a program by declaring a vector and initializing it, and then call function `found()` to determine whether a specific value occurs in the vector.

(b) Write a template version of the function `found()` to determine whether a specific value occurs in a vector of any base type. Test your template function by declaring two or more vectors with different base types and determining whether specific values occurs in these two vectors.

Question 5

Many application programs use a data structure called a dictionary in which one can use a key value to retrieve its associated data value. For example, we might want to associate automobile part numbers with the names of the corresponding parts:

Key	**Value**
100000	tire
100001	wheel
100002	distributor
100003	air filter

The following class interface presents an approach to implementing the above scenario:

```
class Dictionary
{
public:
Dictionary();
     void Add(int key, const string &value);
     string Find (int key)  const;
private:
     vector<int> Keys;
     vector<string> Values;
};
```

The class `Dictionary` has the following operations (member functions):

- `Add()` - adds a new key and value to the dictionary
- `Find()`- retrieves the corresponding value for that particular key, for example `Find(100002)` would return "`distributor`".

Consider the following implementation of the class `Dictionary` and convert it into a template class. In other words, re-design the `Dictionary` interface so that it may be used to create a `Dictionary` containing keys and values of any type. For instance, the value could be of type `double`, whereas the key could be of type `char`. Note the key and value may be most likely of different types hence we need two different template arguments to be supplied.

Also test your template class by declaring two objects of template class `Dictionary` with different template arguments.

Dictionary.h

```
#ifndef DICTIONARY_H
#define DICTIONARY_H
#include <vector>
#include <string>
#include <iostream>
using namespace std;

class Dictionary
{
public:
    Dictionary();
    void add(int key, const string &value);
    string find (int key)  const;
    void display();
private:
     vector<int> keys;
     vector<string>
values;
};
#endif // DICTIONARY_H
```

Dictionary.cpp

```
#include "Dictionary.h"
#include <vector>
#include <iostream>
using namespace std;
Dictionary::Dictionary()
{
    //nothing to do, vector member variables are empty on
    //declaration
};

void Dictionary::add(int key, const string &value)
{
     keys.push_back(key);
     values.push_back(value);
}

string Dictionary::find (int key)  const
{
     string value = " ";
     for (unsigned int i = 0; i < keys.size(); i++)
          if (key == keys[i])
             value = values[i];
     if (value == " ")
         return "no such key can be found";
     else return value;
}

void Dictionary::display()
{
     for (unsigned int i = 0; i < keys.size(); i++)
          cout << keys[i] << ' ' << values[i] << endl;
     return;
}
```

Main.cpp

```
#include <iostream>
#include <cstdlib>
#include "Dictionary.h"
#include <vector>
using namespace std;
int main()
{
     Dictionary parts;
     string part;
     int key;

     //add 4 values to the parts dictionary
     for (int i = 0; i <= 3; i++)
```

```
    {
       cout << "Please enter a part name and a key to add "
              << "to the parts dictionary." << endl;
       cout << "Part name: ";
       getline(cin, part);
       cout << "Key for part name: ";
       cin >> key;
       parts.add(key, part);
       cin.get();
    }
    cout << endl;

    parts.display();
    cout << endl;

    //find the part for a key
    cout << "For which key do you want to find the part? ";
    cin >> key;
    cout << "The part for key " << key  << " is ";
    cout << parts.find(key) << endl;

    return 0;
}
```