C++ Programming Methods Final Exam February 25, 2014

Although this exam is in English, your answers can be in Dutch. Please give the requested code in such way that it could be compiled on a compiler with adheres to the latest ANSI/ISO C++ standard. Be aware of your style and indications. Make sure that variables are defined in the header or as local variable (only exception is the definition of constants). Success!

1 Streams and Strings

Inspect the following code, which makes use of the `stringstream` library:

Listing 1: Stringstream Example Code

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

int main() {
  istringstream is( "aap noot mies" );
  string s;
  is >> s;
  cout << s << endl;
  is.seekg( 0 );
  is >> s;
  cout << s << endl;
}
```

(a) What will be the output of this program (1)? Explain shortly why.

The `stringstream` library is also very usefull for unit-testing, because it can store the test-streams inside the code instead of a number of external test-files.

Listing 2: Stringstream Example Code

```
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
using namespace std;

#define TEST( e )        \
  cout << ((e) ? "Passed: " : "Failed: ") << #e << endl

int Tokenise( istream & ifs , vector <string> & v ) {
  string token;
  while( ifs >> token ) {
    v.push_back( token );
  }
  return v.size();
}

int main() {

  vector <string> v;
  istringstream empty( "" );
  TEST( Tokenise( empty, v ) == 0 );
  TEST( v.size() == 0 );
  v.clear();

  istringstream one( "the" );
  TEST( Tokenise( one, v ) == 1 );
  TEST( v.size() == 1 );
  v.clear();

  istringstream lines( "the quick brown fox\njumped over the lazy dog" );
  TEST( Tokenise( lines, v ) == TEST_PARAMETER_1 );
  TEST( v[5] == TEST_PARAMETER_2);
  v.clear();
}
```

(b) Specify the values of the constants `TEST_PARAMETER_1` and `TEST_PARAMETER_2` in program (2) to make sure that the last two test are passed succesfully.

2 Class inheritance

In this question you are asked to implement class inheritance on some given objects. The main (3) function is given, but the class(es) needed are not. The main object of the problem is the shape, a shape can be a `rectangle` (**height**, **width**) or a `circle` (**radius**). Each `shape` share some common attributes which are the **color** ,the **id** which can be an arbitrary number but always unique, and the **type** which describes the kind of the object. The hierarchy looks like this:

- **Shape** (`int id`, `string color`, `string type`)
  - **Rectangle** (`double height`, `double width`)
  - **Circle** (`double radius`)

The type of each variable is given. In the main function given, three `shapes` are added into a `std::vector`, then the program prints as an output every shape that has been added into the vector with its ID, color, shape (type), attributes, and surface area. All lengths are given in *m*eters and the surface area should be also in the same metric squared.

1. Write C++ code for the class(es) with the appropriate attributes and functions in respect to the code given.
2. What is the output of the main function?

For this question you should use classes and inheritance.

Listing 3: Class Inheritance Main Function

```
int main()
{
    vector<Shape*> shapeList;

    shapeList.push_back(new Rectangle(1, "Green", 2, 2));
    shapeList.push_back(new Circle(2, "Red", 2));
    shapeList.push_back(new Rectangle(3, "Blue", 3, 2));

    for (int i = 0; i < shapeList.size(); ++i)
    {
        cout << "ID: " << shapeList[i]->id <<
                ", Type: " << shapeList[i]->type <<
                ", Color: " << shapeList[i]->color;

        if(shapeList[i]->type == "Rectangle")
        {
```

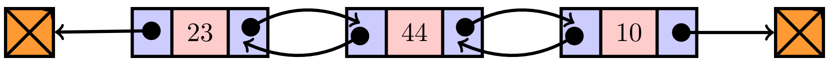
Cont.

```
            cout <<
            ", Height: " << dynamic_cast<Rectangle*>(shapeList[i])->height <<
            ", Width: " << dynamic_cast<Rectangle*>(shapeList[i])->width;
        }
        else
        {
            cout << " Radius: " << dynamic_cast<Circle*>(shapeList[i])->radius;
        }

        cout << ", Area: " << shapeList[i]->area() << endl;
    }
    return 0;
}
```

3 Pointers - Linked Lists

A double linked list consists of `nodes`, each of these nodes has a pointer to the next and the previous nodes. Furthermore a node has a `value` which is an integer in our case. We also hold two pointers to the head(first node) and to the last node(tail) of the list. The pointer to the previous element of the list's head is always `NULL`, the same applies for the next node of the tail node.



Having two pointers in the beginning (**head**) and in the end (**tail**), help us navigate through this list.

(a) Design a class `DoubleLinkedList` with the methods and the data structure that you think is appropriate for each `Node`. (Do **not** implement the functions for insertion, deletion, and searching.)

(b) Give example-code for the function of the class `DoubleLinkedList`, `void insertAtFront(int value)`, which is going to insert a node at the head of the list.

(c) Give example-code for the function of the class `DoubleLinkedList`, `printList()`, which will print all the values of the list starting from the head.

4 Recursion

The Fibonacci sequence $a(1)$, $a(2)$, $a(3)$, ..., $a(n)$, is defined by:
$a(1) = 1$
$a(2) = 1$
$a(n) = a(n-1) + a(n-2), \forall\, n > 2$
This generates the sequence: $1, 1, 2, 3, 5, 8, 13, 21, \ldots$ Write a C++ function `int fibonacci(int n)` that computes the Fibonacci number corresponding to its positive integer argument, so that, for example, `fibonacci`$(7) = 13$.
Extra points: think recursion.

The End.