INFOMGEP 2012	Final exam
Student name:	Student number:
Exam duration: 3 hours	Number of pages: 13 Number of points: 20

All the answers have to be written in the corresponding boxes.
It is allowed to have:

- lecture notes with personal annotations
- books

It is not allowed to have:

- non lecture related hand written material
- any printed material other than the lecture notes
- laptop, calculator and phone

Exercise 1 (2 points)

Draw a box-and-arrow figure of the memory state at the breakpoints in the following program.
Draw a box for each stack variable labeled with name and type, and containing its value when defined.
Draw a box for each object on the heap with its type, and value when defined.
Draw arrows to represent where pointers point to.

```
class Player {};

class Enemy {
public:
   static Player * _thePlayer;
   void setPosition(float * pos) {_position = pos;}
   float * getPosition() const {return _position;}
private:
   float * _position ;
};
Player * Enemy::_thePlayer = NULL;

void main() {
   Player p1; Enemy e1; Enemy * e2 = new Enemy ();
   float pos [3] = {0,1,2}; e1.setPosition(pos);
   if (true) {
      Player p2; Enemy::_thePlayer = &p2;
      // breakpoint 1
   }
   e2->setPosition(e1.getPosition());
   delete e2;
   // breakpoint 2
}
```

At breakpoint 1

stack	heap

At breakpoint 2

stack	heap

Exercise 2 (2 points)

Answer the questions and/or check the answer(s) for which the statement is true. When multiple answers are possible, check all appropriate answers.

Question 2.1

In the following program, what can you write at line `10` to assign the value `1` to `result`?

```
 1  class A {
 2  public:
 3     int data [2];
 4  };
 5
 6  void main() {
 7     A a;
 8     a.data[0] = 0; a.data[1] = 1;
 9     int result;
10
11  };
```

- ☐ `result = a.data[0] + 1;`
- ☐ `result = *(a.data + 1);`
- ☐ `result = *(a.data - 1) + 2;`
- ☐ `result = (--a.data[1])++;`
- ☐ `result = (++a.data[1])--;`
- ☐ `result = a.data[++a.data[0]];`

Question 2.2

Check the answer(s) for which a template is instantiated:

- ☐
```
template <typename T>
T getMax(const T& a, constT& b) { return (a < b) ? b : a;}
```
- ☐ `int a = getMax(10,12);`
- ☐ `float b = getMax<float>(24.1f, 42.0f);`
- ☐ `std::list<int> * numbers;`
- ☐ `numbers = new std::list<int>();`
- ☐ `typedef std::list<int> IntListTypedef;`
- ☐ `#define IntListDefine std::list<int>`

Question 2.3

Considering the following classes:

```
class A {
public:  int valA;
protected:  int valPA;
};

class B : public A {
public:  int valB;
private:  int valPB;
};

class C : protected B {
public:  int valC;
};
```

And assuming that the `main()` function contains the declarations:

```
B b; C c;
```

Check the expression(s) that are allowed in the `main` code (it can be forbidden either because it violates the class definition or the protection mechanism):

☐	b.valA	☐	b.valC
☐	b.valPA	☐	c.valC
☐	b.valB	☐	c.valA
☐	b.valPB	☐	c.valB

Question 2.4

Considering the following program:

```
int getMax(int a, int b) {
   if (a > b) return a;
   if (a < b) return b;
   throw std::invalid_argument("Exception in getMax:  a and b are equal");
};

void main() {
   srand(unsigned int (time(NULL)));
   int a = (int) rand() % 10;
   int b = (int) rand() % 10;
   try { int result = getMax(a,b);}
   catch (std::runtime_error &e) { cout << e.what() << endl;}
   catch (std::logic_error &e) { cout << e.what() << endl;}
   catch (std::invalid_argument &e) { cout << e.what() << endl;}
   catch (std::exception &e) { cout << e.what() << endl;}
};
```

If the exception is thrown in `getMax`, which catch block(s) will be executed?

- ☐ `std::runtime_error`
- ☐ `std::logic_error`
- ☐ `std::invalid_argument`
- ☐ `std::exception`

Exercise 3 (5 points)

As part of your game engine you want to integrate the physics component into the scene manager so that you can directly make use of the data structure used to store the game entities. One important feature of the physics component is to be able to detect collisions between entities. To help you in this task you have programmed the function `void GameEntity::correctIfCollision(GameEntity *)` that corrects the position of two entities colliding (entity on which the function is called and entity in parameter) and ensures that the new positions do not produce collisions with other entities. This function does nothing if the two entities do not collide.

Question 3.1 (2 points)

Your scene manager uses a scene graph to store the game entities. Give the implementation (C++ code) of the two following functions:

`void SceneNode::updateCollision(GameEntity * entity);`
This function corrects collisions between the given entity and the entities attached to the current scene node (and its descendants).

`void SceneNode::updateCollision();`
This function corrects collisions between the entities attached to the current scene node (and its descendants) and every entity in the scene graph. This function can be called on the root node to solve all collisions.

Reminder: A scene graph is a tree structure of SceneNode objects on which GameEntity objects are attached. Therefore each scene node has a container of SceneNode objects called _children and a container of GameEntity objects called _entities. You can access the root node by calling the function SceneNode * getRootSceneNode() on the scene manager.

Question 3.2 (1 point)

A scene graph structure is clearly not the most effective and intuitive structure for that task. Propose a better structure to store the entities checked for collision. Explain (no code needed) how you will implement the collision update function void SceneManager::updateCollision() using that structure.

Question 3.3 (2 points)

In the function `void GameEntity::correctIfCollision(GameEntity *)`, the geometrical representations of the two entities (*e.g.* their bounding box) need to be compared within a common coordinate system. Explain (no code needed) how you would do it and decide whether or not they are colliding. Indicate the information and functions that you would use in each framework (scene graph of Question 3.1 and your own structure of Question 3.2).

Exercise 4 (3 points)

In games, we usually deal with two kinds of light: ambient light and direct light. Ambient light is the light that is present at each point in a 3D scene with the same intensity and apparently coming from all directions with the same intensity. In nature, that kind of light stems from the direct light that is reflected all over the space on every object it hits. Direct light comes from a light source among the following three kinds: directional light, point light and spot light.

The directional light is supposed to come from very distant light sources so that the light rays have already traveled such a long distance that you can think of those you see as being parallel to each other.

A point light source is a single point in space that emits light evenly in all directions.

A spot light is similar to a point light but with the important restriction that it emits light into only a certain angle.

Question 4.1 (1 point)

Give one example of each kind of direct light that you can typically find in a game.

Question 4.2 (2 points)

Propose a class design for modeling the different kinds of lights. Indicate only the class hierarchy and the data members (no function members).

Exercise 5 (5 points)

Usually games are made of levels. When the player completes a level the next one is loaded (old assets deallocated, new ones loaded, timer reset, player's position moved *etc.*) until the last level is completed and then the player wins. Level completion is usually triggered by an entity state update (*e.g.* player position, door opened, enemy killed).

Question 5.1 (1 point)

Give the code or pseudo-code of the function `void GameEngine::nextLevel()` called to manage the level changes. Explain each data and function that you use and indicate in which class they are implemented.

Question 5.2 (4 points)

Before releasing the game that have been created using your game engine, you want to advertise it and make sure that people will like it and buy it. For that you want to create a limited but free version of your game.

Question 5.2.1 (0.5 points)

Limited means here that only the first level of the game will be playable. In that demo mode, the function `nextLevel` will loop back to the beginning after completion of the first level. Give the code or pseudo-code of the new version of `nextLevel`.

Question 5.2.2 (0.5 points)

The version that you have proposed in the previous question probably still exports the code related to the full game, *e.g.* the loading of the other levels. It means that one could edit the executable file, change the few values corresponding to the test if you are in demo mode and make a free full version of the game, ruining your business plan...

Propose a solution to prevent such reverse engineering of the game (no code needed), *i.e.* how you can make sure that addition of functionalities cannot be done by edition of the executable.

Question 5.2.3 (1.5 points)

You now want to release an even more limited version of your game where the player cannot interact with the game at all. The input manager is basically disable except for pausing the game and stopping the program. The game is playing by itself, and the player is just watching it. You are thinking of two ways to achieve this.

One way is to start up the first level as normal, and playback pre-recorded player's key strokes. The other one is to implement an AI that will simulate what the player might do in the same circumstances.

List the advantages and disadvantages of each method.

Question 5.2.4 (1.5 points)

You finally come up with another method which consists in recording the entire game state at each game loop iteration and then playing it back in demo mode. This time you do not record the key strokes but the entire game state, *i.e* in our case the scene graph. Compare this method to the other two and explain how you would implement it (no code needed), and how you could optimize it.

Exercise 6 (3 points)

One useful feature in a game engine is to be able to maintain a list of the highest scores. As for online games you cannot store that list locally, you decide to use a dedicated server that stores and updates the highest scores. To manage high scores in your game engine, you design the following `ScoreManager` class:

```
class ScoreManager {

private:
     std::string _serverName;
     // DNS name of the server

     std::string _portNumber;
     // Port number of the server

     int _sock;
     // The socket to communicate with the server

     std::vector<std::pair<int,std::string>> _highScores;
     // The highest scores made of pairs (score , player's name)
```

```
    void connectToServer();
    // Establishes the communication, called by constructor

    void disconnectFromServer();
    // Disconnects from the server, called by destructor

public:
    ScoreManager(std::string server, std::string port);
    // Constructor

    ~ScoreManager();
    // Destructor

    void updateHighScore();
    // Asks the server for highest scores and updates _highScores

    bool submitScore(int myScore, std::string myName);
    // Submits the player's score, returns submission successfulness

    int getScore(int index) const ;
    // Returns the index-th highest score

    std::string getPlayerName(int index) const ;
    // Returns the player's name of the index-th highest score
};
```

Question 6.1 (1 point)

Indicate what network protocol you would use and why.

Question 6.2 (1 point)

Give the code or pseudo-code of the function:

`bool submitScore(int myScore, std::string myName);`

Question 6.3 (1 point)

Explain how to hide the private data members of `ScoreManager` to other classes. Illustrate your answer with code or pseudo-code.