Task 1:

1. Create a class *CComputer*.

2. The class *CComputer* has the following structure:

```
class CComputer{
public:
	void Display();			// print all the information about a computer
	CComputer (/*parameters*/); // constructor
	CComputer ();			// default constructor
	~CComputer ();			// destructor
private:
	double m_uProcessorFrequency;	// processor frequency in GHz;
	unsigned m_uHD;			// HD size in GB;
	unsigned m_uRAM;			// RAM size in GB;
};
```

3. Create a *main* function, which allows testing objects of *CComputer* type, for instance:

```
int main (int argc, char* argv[])
{
	CComputer c1(2.4, 1000, 16);
	CComputer c2(2.2, 500, 8);
	CComputer c3;
	c1.Display(); c2.Display(); c3.Display();
	return 0;
}
```

Task 2:

1. Create a class *CStaticStack* which represents a fixed size stack of computers (the size is defined in the constructor). Implement the stack as an array of objects.

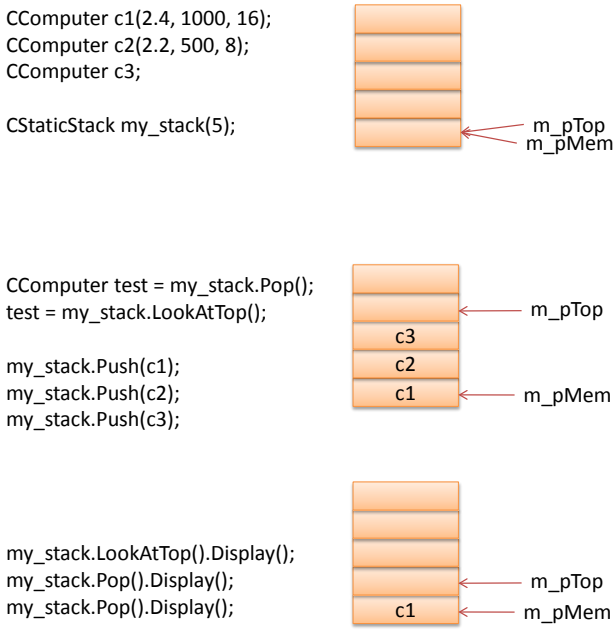
2. The class *CStaticStack* has the following structure:

```
typedef CComputer stack_item;
class CStaticStack
{
public:
	stack_item Pop();
	void Push(stack_item);
	stack_item LookAtTop();
	bool IsEmpty();

	CStaticStack (unsigned size);
	~CStaticStack();

private:
	unsigned m_uSize;
	stack_item* m_pMem;
	stack_item* m_pTop;
};
```

3. Design a program which exemplifies the use of the stack of computers.



Task 3:

1. Create a stack of integers *CStack* which allows eliminating the size restrictions. Organize the stack as a linked list of nodes, storing within each node both an integer and a pointer to the next node.

```
typedef int StackItemType;
struct StackNode;

class CStack
{
public:
       // constructor and destructor
       // methods
private:
    StackNode* m_pTop;      // pointer to the first stack node
};
----------------------------
struct StackNode            // a stack node
{
  StackItemType item;       // data to store in the stack
  StackNode* pNext;         // pointer to the next node
};
```

2. Use the constructed stack in order to convert a positive integer number represented in decimal to a binary number (to a sequence of 0s and 1s stored in the stack).