Introduction

The aim of this proficiency test is to discover whether the student's programming skills in C/C++ language are enough to undergo the current course.

The test is compulsory for students who are planning to gain the Master's degree at our University but have not passed the examination in subject IAG0090 ("Algorithms and Data Structures").

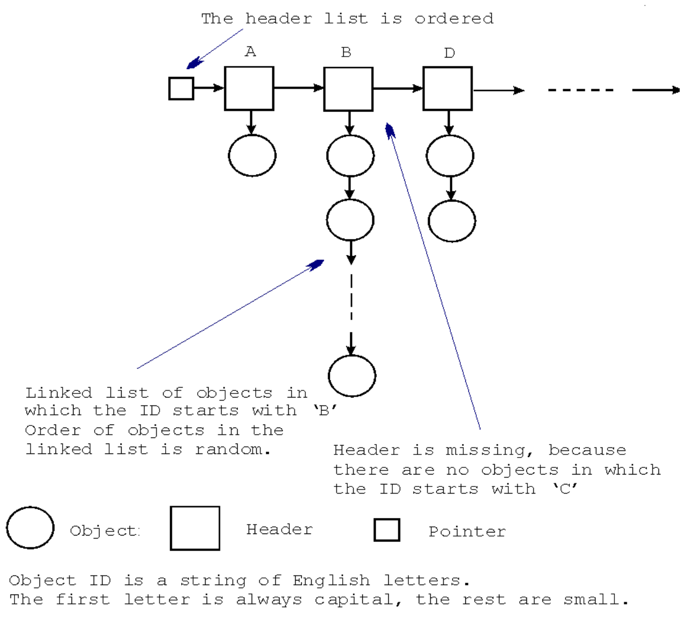
To pass the test, the student has to write on its own laptop computer a small C Windows console application and demonstrate to the instructor that it operates properly. The evaluation will take place on the second week of semester right after the lecture.

Students who have not passed the test cannot participate this course. However, it is not a catastrophe. Learn to code in C language and come back on the next autumn. Participating in the "Programming I" (IAX0583, consult Ass. Prof. Vladimir Viies) course may help you.

For guest students the test is recommended but not compulsory. The test simply helps you to make clear for yourself, is the "Advanced Programming" course manageable for you or not. If you are able to develop the small C application specified in this test within two or three hours, you should not worry – you will manage. However, if this test is too overwhelming for you or if the solution demands enormous efforts from you or if you need very significant help from your friends, then it is wise to select some other subject. On the other side, if you think that the current course is too simple and you cannot gain any knew knowledge from it, turn your attention to "Embedded Systems" (IAY0330, consult Prof. Thomas Hollstein or Prof. Kalle Tammemäe) or to "Real-time Operating Systems and Syetem Programming" (ITI8510, consult eng. Jaagup Irve).

Initial data

The *StructGenerator* function delivered by the instructor creates a linked list that can be illustrated with the following example:


The header list is ordered
A
B
D
Linked list of objects in which the ID starts with 'B' Order of objects in the linked list is random.
Header is missing, because there are no objects in which the ID starts with 'C'
Object
Header
Pointer
Object ID is a string of English letters.
The first letter is always capital, the rest are small.

Here:

```
typedef struct ob
{
	char *pID;
	unsigned long int Code;
	char *pTime; // formatted by string "%02d:%02d:%02d"
	             // for example, the result may be "15:52:30"
	struct ob *pNext;
} Object;
typedef struct head
{
	Object *pObject; // Pointer to the linked list of objects.
	char cBegin;     // The linked list contains objects in which
	                 // the first letter of object ID is cBegin.
	struct head *pNext;
} Header;


Header *StructGenerator(int nObject); // nObject is the number of objects
```

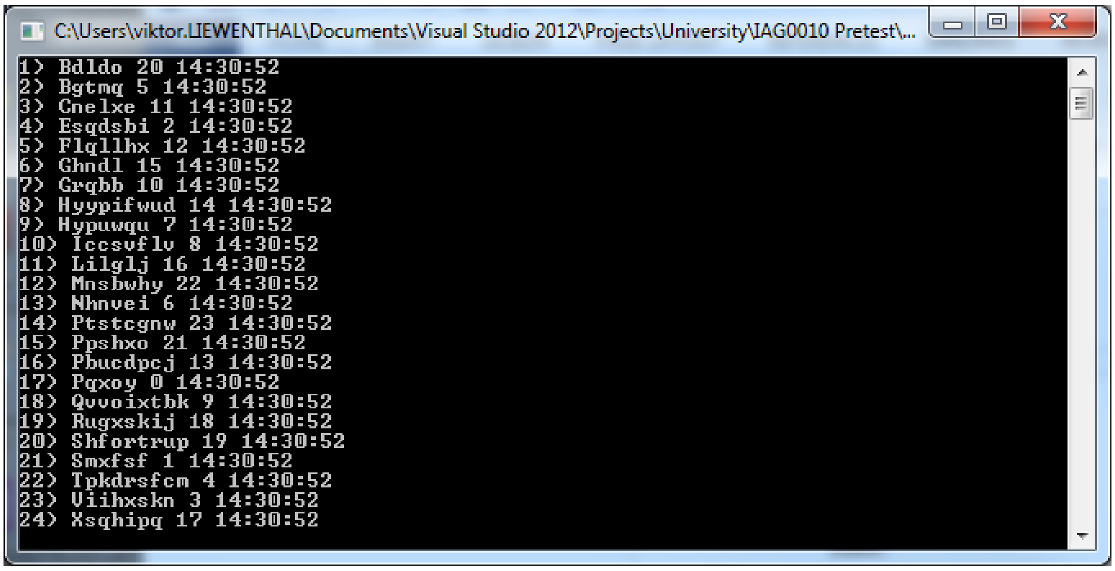
The *pID* is the pointer to a C string. Its number of characters may be any value except 0. The *Code* may be any integer.

Let us emphasize that the list presented on the figure above is just an example. The *StructGenerator* function uses random values and the actual list may look like in rather different way (for example, objects in which the ID starts with 'A' may be missing, some objects in which the ID starts with 'C' may exist, etc.).

Application you have to develop

The C/C++ console application you have to develop must consist of three functions:

1. Function with prototype
   *void PrintObjects(Header *pList);*
   that prints the contents of all the objects from the *StructGenerator* created list into Command Prompt window. *pList* is the output value of *StructGenerator*.
   Example of the printout:


C:\Users\viktor.LIEWENTHAL\Documents\Visual Studio 2012\Projects\University\IAG0010 Pretest\...

```
1) Bdldo 20 14:30:52
2) Bgtmq 5 14:30:52
3) Cnelxe 11 14:30:52
4) Esqdsbi 2 14:30:52
5) Flqllhx 12 14:30:52
6) Ghndl 15 14:30:52
7) Grqbb 10 14:30:52
8) Hyypifwud 14 14:30:52
9) Hypuwqu 7 14:30:52
10) Iccsvflv 8 14:30:52
11) Lilglj 16 14:30:52
12) Mnsbwhy 22 14:30:52
13) Nhnvei 6 14:30:52
14) Ptstcgnw 23 14:30:52
15) Ppshxo 21 14:30:52
16) Pbucdpcj 13 14:30:52
17) Pqxoy 0 14:30:52
18) Qvvoixtbk 9 14:30:52
19) Rugxskij 18 14:30:52
20) Shfortrup 19 14:30:52
21) Smxfsf 1 14:30:52
22) Tpkdrsfcm 4 14:30:52
23) Viihxskn 3 14:30:52
24) Xsqhipq 17 14:30:52
```

2. Function with prototype
   *int RemoveObject(Header **ppList, char *pExistingID);*
   that finds the object with *pExistingID* from the *StructGenerator* created list, deletes it and rearranges the linking pointers. The return value is 0 if the object was not found, otherwise the return value is 1. Turn attention to the first parameter: it is the pointer to pointer. The reason of that is as follows. If there is only one object in a chain and you remove it, you must remove also the corresponding to it header. But if the header you remove is the first one, the value of pointer that points to the beginning of structure (i.e. the output value of *StructGenerator)* must get also a new value.
3. The main function that calls the *StructGenerator* and the other two functions.

How to start

You need Microsoft Visual Studio (any edition starting from year 2013). The Visual Studio Express Edition (called also as Visual Studio Community, see https://www.visualstudio.com/en-us) is good enough.

1. Start Visual Studio.
2. Select *File → New → Project*. From the *New Project* dialog box select *Visual C++ → Empty Project*. Set the project name (for example, *Pretest)* and folder and click *OK*.
3. Select *Project → Add New Item.* From the *Add New Item* dialog box select *C++ File.* Set the new file name (for example, *Pretest.cpp*) and click *Add.*
4. Select *Project → Add New Item.* From the *Add New Item* dialog box select *Header File.* Set the new file name (to avoid later contradictions, choose *Pretest.h*) and click *Add.*
5. Copy the instructor's file *StructGenerator.cpp* into the folder where your two just created files are.
6. Select *Project → Add Existing Item.* From the *Add Existing Item* dialog box select *StructGenerator.cpp* and click *Add.*
7. Into your new header file type the declarations of structs *Object* and *Header* and function *StructGenerator*.
8. Into your new source file type the following text:

```
#include "Pretest.h"
#include "stdio.h"

#pragma warning (disable : 4996)

int main()
{
        Header *pList = StructGenerator(25);
        printf("Hello\n");
        return 0;
}
```

9. Select *Build → Build solution.*
10. Select the last row of your source code and press *F9*. You will get the breakpoint which stops your program right before the end.

11. Select *Debug* → *Start debugging.* You should see the Command Prompt window with text *Hello*.
12. Select *Debug* → *Stop debugging.*

Now your preparations are completed and you may start to write your own code.