**C++ Workshop – 150018**

**Homework Assignment #9**

**STL**

**Question # 1:**

This exercise deals with medals given after the operation "Guardian of the Walls ".

Privates, commanders, and officers took part in operation "Guardian of the Walls ".

Criteria were established for receiving a medal for each position:

• Private: Participated in at least 10 military operations and average scores of his participation in each operation higher than 95

• Commander: Participated in at least 7 military operations and the average scores of his participation in any operation higher than 90, in addition he is in a combat unit

• Officer: Participated in more than 2 military operations and received a sociometric score from the soldiers of at least 92

To represent these roles, you must define a hierarchy of departments as follows:

1. An abstract base class named Soldier for representing a soldier.  
   Fields of the class include: ID number, first name, last name, number of operations in which the soldier participated.  
   Add the following methods to the class:
   1. **Constructor** to initialize the data members
   2. **virtual destructor**
   3. **medal()** - a Boolean method that checks whether the soldier is entitled to a medal
   4. **print()** for printing soldier data. The soldier's data must be printed according to his attributes. When printing, first print the attribute name and then the attribute values, each attribute is on a separate line.

B. Private class inheriting from Soldier for representing a privates, which contains, in addition to the personal details, also a pointer to an array of evaluation scores of the military operations in which he participated (score range 0-100). Note, this is a class with dynamic fields that requires the application of deep copying in the relevant methods.

C. A Commander class that inherits from Private for the representation of a commander, which contains, in addition to the existing data members, a Boolean attribute that indicates whether the commander is combat or not.

D. An Officer Class that inherits from Soldier for representing an officer, which contains, in addition to the personal details, a field for his sociometric score.

Note, you need to set up the classes correctly, define virtual methods, pure virtual, etc.

How to print the data for each soldier is described as follows:

Commander

ID

first name:

last name:

num operations:

grades: print scores array with blanks in between

combat: yes/no

Officer

ID

first name:

last name:

num operations:

sociometric score

Private

ID ת.ז

first name:

last name:

num operations:

grades: print scores array with blanks in between

E. Add to each of the above classes a method called **soldierType()** which **returns a string** indicating the type of soldier. (The method will return "private", "commander", "officer" according to the different types of soldiers)

F. In the main program, we will define a vector or list (your choice) using an STL container which can accommodate soldiers **of all types**. In addition, you must define a number of global functions that operate on a vector or list of soldiers (of all types) as follows:

* add function that receives a vector or list (of STL) of soldiers and adds a new soldier to its program.

For this function, use only the following outputs:

cout<<"enter 1 to add a private soldier\n";

cout<<"enter 2 to add a commander soldier\n";

cout<<"enter 3 to add a officer soldier\n";

cout<<"enter id, first name, last name and number Of operations\n";

cout<<"enter "<<numOfOperations<<" grades\n";

cout<<"enter 1 if the soldier is combat and 0 if not\n";

cout<<"enter number of sociometric score\n";

* + A function **medal()** that receives a vector or a list of soldiers and prints out only the details of the soldiers who are eligible for a medal.
  + **mostSociometricScore()** function that receives a vector or list of soldiers and returns a pointer to a soldier on the list whose position is an officer and whose sociometric score is highest. If there are no officers on the list the function will return NULL.

**Important Note**: For implementing this function, a method must be defined that returns the sociometric score of the officer and exists only in the Officer Department. Because the soldiers are in a collection that contains Soldier-type objects, it is not possible to invoke methods that are not defined in the Soldier class. Therefore a virtual method should be added to the Soldier class. The method should not be defined as pure virtual in order not to require derived classes to implement it. If it is not supposed to be called, it should throw an error:   
"ERROR: this function is just for officer soldier"

* + In the main program you create an STL vector or list list (of your choice) which contains pointers to soldiers of **any types** (private, commander, officer).
  + The user will then be shown an action to perform (in a loop) until STOP is selected.  
    Possible actions:

0 - End of program

1 - Adding a new soldier

2 - Printing the data of the soldiers who are entitled to the medal.

3 - Printing the name (family and first name) of the soldier in the position of officer with highest sociometric score

4 - Printing the number of private soldiers eligible for the medal

5 - Printing the names (family and first names) of the officers who are not in a combat unit.

6 - Print message If there is a soldier on the list who has participated in more than 15 operations

7 - Deletion from the vector / list of soldiers and officers who did not participate in any operations

Given the following main program, you must complete the missing lines of code. Note that wherever the underscore is marked, only one line must be completed (not necessarily according to the length of the line). Use lambda expressions and call the algorithmic functions defined in the STL algorithm library.

Note: you can get help from the STL in the following site:

https://www.cplusplus.com/reference/

#include <iostream>

#include <list>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

enum option{

|  |  |  |
| --- | --- | --- |
| stop, | // | End program |
| addNewSoldier, | // | Add new soldier |
| medalList, | // | Print all soldiers that are entitled to a medal |
| mostSociometric, | // | Most sociometric score |
| countMedalPrivate, | // | Number of private soldiers getting a medal |
| noCombatCommander, | // | Names of officers not combat |
| overSoldier, | // | A message for soldier participating in more than 15 operations |
| removeOfficer, | // | Removing officers that did'nt participate in any operation |

};

void add(\_\_\_\_\_\_\_\_\_ ); //vector or list

void medal(\_\_\_\_\_\_\_\_\_ ); //vector or list

Soldier\* mostSociometricScore ( \_\_\_\_\_\_\_\_\_); //vector or list

int main()

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // define vector or list of soldiers

int op;

cout<<"enter 0-7\n";

cin>>op;

while(op!=stop)

{

switch (op)

{

case addNewSoldier:add( \_\_\_\_\_\_\_\_\_\_\_\_ ); //add new soldier break;

case medalList:medal( \_\_\_\_\_\_\_\_\_\_\_\_ ); //print entitled for a medal

break;

case mostSociometric:

Soldier\* s=mostSociometricScore( \_\_\_\_\_ ); // vector or list

cout<<"Officer soldier with most sociometric score: ";

cout <<s->getFname()<<' '<<s->getName()<<endl;

break;

case countMedalPrivate: cout<<"# private soldier for medal: ";

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //number of privates entitled for medal

cout<<endl;

break;

case noCombatCommander: cout << "list of no combat commander soldier :";

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cout<<endl;

break;

case overSoldier:

if( \_\_\_\_\_\_\_\_\_\_\_\_\_\_ )

cout<<"there is a soldier that takes more than 15 operations\n";

else cout<<"no soldier takes more than 15 operations\n";

break;

case removeOfficer:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //print list after removals

break;

};

cout<<"enter 0-7\n";

cin>>op;

}

return 0;

} }

Example:

enter 0-71

enter 1 to add a private soldier

enter 2 to add a commander soldier

enter 3 to add a officer soldier

1

enter id, first name, last name and number Of operations

111 aaa aaa 3

enter 3 grades

100 95 98

enter 0-7

1

enter 1 to add a private soldier

enter 2 to add a commander soldier

enter 3 to add a officer soldier

2

enter id, first name, last name and number Of operations

222 bbb bbb 0

enter 1 if the soldier is combat and 0 if not

1

enter 0-7

1

enter 1 to add a private soldier

enter 2 to add a commander soldier

enter 3 to add a officer soldier

3

enter id, first name, last name and number Of operations

333 ccc ccc 0

enter number of sociometric score

100

enter 0-7

3

Officer soldier with most sociometric score: ccc ccc

enter 0-7

4

# private soldier for medal: 0

enter 0-7

5

list of no combat commander soldier :

enter 0-7

6

no soldier takes more than 15 operations

enter 0-7

7

private

ID: 111

first name: aaa

last name: aaa

num operations: 3

grades: 100 95 98

ID: 222

first name: bbb

last name: bbb

num operations: 0

grades:

combat: yes

enter 0-7

0