**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.

**Note:**  for the **grades** attribute, if there are no grades then do not print the word : *grades*

* 1. **Getters and setters for all data members.**
  2. **soldierType** a method that will return a string value based on the type of solder that the soldier is. (The method will return either the word **private** or **commander** or **officer**.)

B. **PrivateSoldier** 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 **PrivateSoldier** 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, including correct definitions for constructors (including copy and move constructuors) correct definitions for virtual methods, pure virtual methods, etc. There is no need to define set functions.

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

PrivateSoldier

ID ת.ז

first name:

last name:

num operations:

grades: print scores array with blanks in between

Note: each of the inheriting classes will have to override the method **soldierType()**

E. 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 the vector (or list)

For this function, use only the following outputs:

cout << "choose a soldier\n";

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

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

cout << "enter 3 to add an officer\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 the 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.
  + **highestSociometricScore()** 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 such officers on the list the function will return NULL.

**Important Note**: In order to implement this function you will need to call the method **getSocioMetric()**  that **only** exists in the **Officer**’s class. Since the vector (or list) is defined to contain pointers to objects of type **Soldier** this will cause a compilation error. (During compilation time, if there is a method that is being called on an object that supposedly holds a pointers to Soldier, it will check if the method exists in the class Soldier and if not then it will create a compilation error. The compiler does not know or care that at run time it would be an Officer) In order to get around this problem we will do the following

1. Before we call the method **getSocioMetric** we will first make sure that the soldier is indeed of type **Officer**.
2. We will cast the object to be of type **Officer** so that we will not have a compilation error.

Aka, if we had the following definition:

Soldier \*soldier = new Officer();

Then we would write the following code:

if (soldier->soldierType()==”officer”)

((Officer \*) soldier)->getSociometric();

* + 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 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 {

EXIT, // סיום התוכנית

ADD\_NEW\_SOLDIER, // הוספת חייל חדש

DESERVES\_MEDAL, // הדפסת פרטי כל החיילים הזכאים לצל"ש

HIGHEST\_SOCIOMETRIC, //הדפסת שם (משפחה ופרטי) של החייל בעל ציון סוציומטרי מקסימלי

PRIVATE\_MEDAL\_COUNT, // הדפסת מספר החיילים הטירוניים הזכאים לצל"ש

NONCOMBAT\_COMMANDER, // הדפסת שמות (משפחה ופרטי) של המפקדים שאינם בקרבי

SUPER\_SOLDIER, //הדפסת הודעה מתאימה, האם קיים חייל שהשתתף ביותר מ- 15 מבצעים צבאיים

REMOVE\_OFFICER, // מחיקת כל החיילים הקצינים שלא השתתפו כלל במבצע צבאי

};

void add(\_\_\_\_\_\_\_\_\_ ); //השלם\י פרמטר- ווקטור או רשימה

void printMedalList(\_\_\_\_\_\_\_\_\_ ); //השלם\י פרמטר- ווקטור או רשימה

Soldier\* highesttSociometricScore ( \_\_\_\_\_\_\_\_\_); //השלם\י פרמטר- ווקטור או רשימה

int main()

{

Soldier\* s;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // הצהרה על ווקטור או רשימה של חיילים

int op;

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

cin>>op;

while(op!=EXIT)

{

switch (op)

{

case ADD\_NEW\_SOLDIER:add( \_\_\_\_\_\_\_\_\_\_\_\_ ); //הוספת חייל חדש

break;

case DESERVES\_MEDAL:printMedalList( \_\_\_\_\_\_\_\_\_\_\_\_ ); //הדפסת פרטי הזכאים לצל"ש

break;

case HIGHEST\_SOCIOMETRIC: //הדפסת שם הקצין בעל ציון סוציומטרי גבוה ביותר

s=highestSociometricScore( \_\_\_\_\_ ); // השלם\י פרמטר-וקטור או רשימה

cout<<"Officer with the highest sociometric score: ";

cout <<s->getFirstName()<<' '<<s->getLastName()<<endl;

break;

case PRIVATE\_MEDAL\_COUNT: cout<<"number of privates that received medals: ";

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //הדפסת מספר הזכאים לצל"ש בטירונים

cout<<endl;

break;

case NONCOMBAT\_COMMANDER: cout << "list of noncombat commanders: "; //הדפסת רשימת(שם משפחה ופרטי) החיילים המפקדים שאינם בקרבי

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

cout<<endl;

break;

case SUPER\_SOLDIER:

if( \_\_\_\_\_\_\_\_\_\_\_\_\_\_ ) // קיים חייל שהשתתף יובתר מ- 15 מבצעים צבאיים

cout << "there is at least one soldier that did more than 15 operations\n";

else

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

break;

case REMOVE\_OFFICER://מחיקה מהווקטור/רשימה של כל החיילם קצינים שאינם השתתפו כלל במבצעים צבאיים

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //הדפסת כל הרשימה לאחר מחיקת האיברים

break;

};

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

cin>>op;

}

return 0;

}

Example:

enter 0-7

1

choose a soldier

enter 1 to add a private

enter 2 to add a commander

enter 3 to add an officer

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

choose a soldier

enter 1 to add a private

enter 2 to add a commander

enter 3 to add an officer

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

choose a soldier

enter 1 to add a private

enter 2 to add a commander

enter 3 to add an officer

3

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

333 ccc ccc 0

Enter the sociometric score

100

enter 0-7

3

Officer with the highest sociometric score: ccc ccc

enter 0-7

4

number of privates that received medals: 0

enter 0-7

5

list of noncombat commanders :

enter 0-7

6

no soldier did more than 15 operations

enter 0-7

7

private

ID: 111

first name: aaa

last name: aaa

num operations: 3

grades: 100 95 98

commander

ID: 222

first name: bbb

last name: bbb

num operations: 0

combat: yes

enter 0-7

0

**Good Luck!**