CSI 2372[A] – Fall 2021

Assignment 3

Due Date: 12th November, 11:00 PM

Question 1 of 2) **Person and Organization Classes**

For this Assignment, you need to create 2 classes: **Person** (represents a person) and **Organization** (represents an organization).

People can join an organization, and each organization must maintain a table of all registered members.

The Person class has two private members: The person's name and age. (You can give default values for name and age)

```
std::string name;   // person's name
int age;            // person's age
```

You must create public access methods for these 2 members.

```
int getAge();
std::string getName();
```

The Organization class has these private members:

class Organization {

```
std::string name; // name of the org
Person* members; // list of members
int size;  // actual size of the org
int dim;  // max size of the org
```

- **name**: the name of the organization (you can give a default value for name)
- **members**: a pointer to an array of Persons
- **size**: the actual size of the members array (the actual number of people in the organization)
- **dim**: the maximum size of the members array (the maximum number of people the organization can have). The default size of an organization is **100 Persons.**

The Organization class has a public method to return the list of members of the organization:

```
// return the name of all members entolled in the organization
// all members written in onre string separated by semi-colons
// returns empty string if no member is enrolled in the organization
std::string getMemberNames()
```

The Organization class also has another public method to add a person to the array of members of the organization (We double the max size of the array if the array is full):

```
// adds a Person to the members array of the organization
// Should double the members array size if array is full
void addPerson(Person person)
```

Note: the array of members of an Organization must be dynamic, so be sure to add the necessary methods for properly managing dynamic memory (constructors, destructors etc...)

You provide at the end a global void function to implement (**printMembers()**) to display the members of an organization at std::cout:

```
void printMembers( Organization organization)
```

At the end, you are given a main() program to test your classes.

You can't modify the principal program, so make sure you don't get any runtime or compilation errors.

Question 2 of 2) **Inheritance (Specialization, Generalization)**

We will take the Organization and Person classes from Question 1 and add a sub-class and some dependencies to them. The main() code file is available in the BS.

First, you need to add an **array of pointers** to organizations in the **Person** class**.**

```
Organization** organizations;
```

A person can be member of at most 5 organizations (fixed size array), so `organizations` attribute is a pointer to a dynamic array of type `Organization *` and size 5. If we try to add a 6th organization, then an exception std::out_of_range is thrown. The constructor of **Person** class is a good function to to this initialization.

The `registering` non-member function that will allow to add an organization to a person and a person to an organization with the following format:

```
//add a person to an organization and an organization to a person
void registering(Organization* o, Person* p) {
    // two lines of code
}
```

The **addPerson** function has already been implemented in Question1, so for the `registering` function you need to call **addPerson** in the **Organization** class and you have to call the **addOrganization** function in the **Person** class.

```
void Person::addOrganization(Organization* organization)

// adds a Person to the members array of the organization
// Should double the members array size if array is full
 void Organization::addPerson(Person person)
```

Then, you must add a sub-class of Organization called **University** which will have an attribute **tuition** (float) that you can access with getTuition() and setTuition().

The `getOrgNames() of Class` **`Person`** `returns a string of all the organizations of the person.`

```
//returns a string containing all the organizations (org and univ) of the person
       std::string Person::getOrgNames()
```

Add a method `getTotalTuition()` method in the **Person** class that will sum the Tuition for universities that a person is registered. Some Person class member function prototypes:

```
       float Person::getTotalTuition()

//returns a string containing all the universities of the person

      std::string Person::printSchools()
```

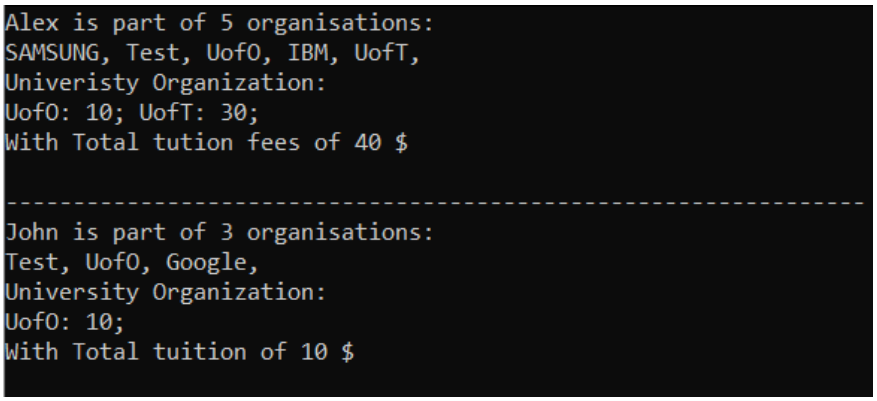
You have to loop through the **`organizations`**, do a **downcast (**with **dynamic_cast)** in order to check which elements that are pointed by the **`organizations`** are University objects and sum up the results of getTuition(). The pointers in this dynamic array are accessible by [].

> **dynamic_cast** allows you to determine if a pointer is pointing to a derived class (University) or to a base class (Organization), which is specified by the return value of dynamic_cast.

Add the ordinary non-member function **printMembers** function that prints the name of the person, the number of organizations the person belongs to and the set of organizations the person belongs to.

```
//prints the name of the person, the number of organizations of the person and the name
list of the organizations of the person
void printMembers(Person& person)
```

The output should look like this:



Submit your assignment to Brightspace as a zip file. Your files must have your name and student number.

Restrictions:

- The program must have ONLY the following classes and subclasses: Organization, Person and University
- It is possible to add header and .cpp files to design the classes.
- The main() program should not change, but the Organization and Person class should.
- You can use your Organization and Person classes implemented in Question 1 to make Question 2
- You can add any getters/setters you want to design the program.