

NYU – Tandon School of Engineering

Brooklyn, NY 11201

# CS-UY 2124 / OOP

Midterm Exam – 12 June 2020

Prof. Katz

Name: \_\_\_\_\_

NetID(first part of email): \_\_\_\_\_

- **YOU WILL TAKE THIS EXAM ON GRADESCOPE!!!**
- Please submit ONLY 3 files (one for each question)
- You can expect that the user inputs the appropriate values (int/float/etc where required).
- Comments are not required
- You may use your notes, your compiler and the Internet however you may not use any assistance from any other person (no Chegg resources, homework helper or another student)
- Anyone found cheating on this exam will receive a zero for the exam.
- If you have a question please email the professor (not the TAs)!

1. Design a set of classes to model a phone book. Each Person entry in the phone book should contain a person's name (string), and multiple telephone numbers. The name is mandatory and cannot be changed. Each telephone number will contain a string to indicate the type of number (Mobile, home, work, etc) and an integer for the actual phone number (our system has simple numbers like 1234 or 2112, that's all). Your phone book class will store all of the people. You must make all data members private!

The classes should have the following functions:

- The PhoneBook class
  - i. A function to add a new person, with one new phone number and type
  - ii. A function to search the phone book for a person given their name, returns type Person (the person WILL exist)
  - iii. A function to search the phone book for a phone number, returns type Person (the person WILL exist)
- The PhoneNumber class
  - i. Accessors and Mutators (Getters and Setters) for both items
- The Person Class
  - i. A function to add a new phone number
  - ii. A function to search for a particular type of phone number, returns int if found or -1 if not found.

Notes:

- Do not use any STL classes we have not discussed in this course
2. Given a file with the GPA (double) of all students, determine the name of the student with the highest GPA. The filename will be given to you by the user (be sure to check for validity), and is in two columns, where the first column holds the student's GPA, and the second holds their name, a tab character is used to separate the two columns. Name can be any number of individual words separated by spaces.

A sample of the file is below:

```
2.5    John Jones
4.0    Madonna
3.773  Ulysses S. Grant
```

(The resulting output of the above file would be "The valedictorian is: Madonna")

3. An animal shelter has a lot of animals for adoption and would like to be able to identify the people who want to adopt those animals. Additionally, it would be helpful if we could identify those animals which no one has expressed an interest in adopting so that their prominence on the website can be increased (thereby increasing their likelihood of adoption). Also, we'd like to find out which people in our system have not, yet, adopted any pets so we can send them a message!

For this problem, you will design two classes, Pet and Adopter. Each Adopter can adopt only one Pet however each Pet can be adopted by multiple adopters (multiple people are interested in adopting the same pet). Your task is to identify the potential adopters (people who have not adopted a pet) and which pets have no adopters.

Hints:

- For this solution, you will need the Pet class to be friends with the Adopter class (put "friend class Adopter;" inside the Pet class)
- Define your longer functions OUTSIDE of the class and after all class definitions

The Adopter class will have:

- The person's name
- a pointer to the Pet which that person is interested in adopting.
- A constructor which could be provided, but does not require, a name for the Adopter
- An accessor and mutator (getter and setter) for the name.
- A function "isAdopting" which returns true if the adopter has selected a Pet to adopt
- A function "getPetName" which returns the name of the pet they want to adopt (a string)
- A function "adopt" which will take a Pet by reference and adjust both objects accordingly and returns true if the adoption process was allowed, false if not. (remember, you should not allow a person to adopt more than one Pet!)

The Pet class will have:

- The Pet's name
- A vector of pointers to the Adopters which are interested in that pet
- A constructor which could be provided, but does not require, a name for the Pet
- An accessor and mutator (getter and setter) for the name.
- A function "numberOfAdopters" which returns the number of potential Adopters
- A function "getAdopterNames" which returns a vector (by value) with the names of the potential adopters

(sample Main and run of the program on the next page)

```

int main() {
    Adopter betty("Betty");
    Adopter tom("Tom");
    Adopter alice("Alice");
    Adopter bob("Bob");
    Pet fido("Fido");
    Pet fluffy("Fluffy");
    Pet smellyCat("SmellyCat");

    if (betty.adopt(fido))
        cout << "Betty Adopted Fido" << endl;
    else
        cout << "Betty cannot adopt Fido" << endl;

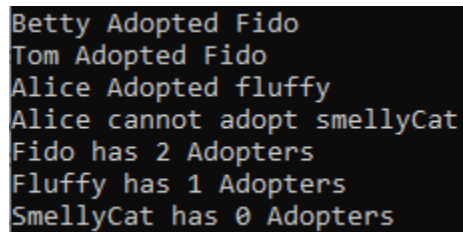
    if (tom.adopt(fido))
        cout << "Tom Adopted Fido" << endl;
    else
        cout << "Tom cannot adopt Fido" << endl;

    if (alice.adopt(fluffy))
        cout << "Alice Adopted fluffy" << endl;
    else
        cout << "Alice cannot adopt fluffy" << endl;

    if (alice.adopt(smellyCat))
        cout << "Alice Adopted smellyCat" << endl;
    else
        cout << "Alice cannot adopt smellyCat" << endl;

    cout << "Fido has " << fido.numberOfAdopters() << " Adopters" << endl;
    cout << "Fluffy has " << fluffy.numberOfAdopters() << " Adopters" << endl;
    cout << "SmellyCat has " << smellyCat.numberOfAdopters() << " Adopters" << endl;
}

```



```

Betty Adopted Fido
Tom Adopted Fido
Alice Adopted fluffy
Alice cannot adopt smellyCat
Fido has 2 Adopters
Fluffy has 1 Adopters
SmellyCat has 0 Adopters

```