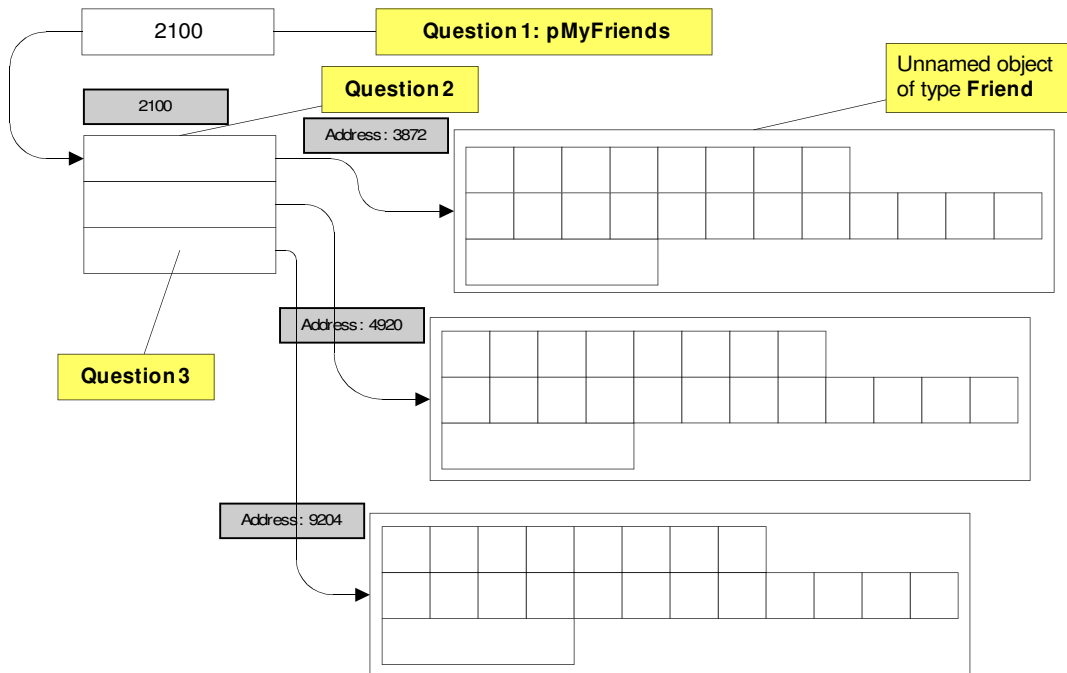


NET2006: Object-Oriented Programming: Fall 2008: Midterm: Solution

Part 1: Answer all questions first on this paper. Once you have finished answering, transcribe your answers to the green mark-sense cards, using only a pencil. You will submit your mark sense cards, but keep the quiz paper. Following the lecture, you can check your answers with those posted on my website.

Following the completion of **Part 1**, submit your answers. You can then take an unsupervised break before you begin **Part 2**.



The following 3 questions, are based on the diagram shown above.

- Identify the declaration / definition of **pMyFriends**:
 - `Friend* pMyFriends[];`
 - `Friend* pMyFriends[numFriends];`
 - `Friend* pMyFriends*;`
 - `Friend** pMyFriends;`
 - `Friend* pMyFriends;`
- Identify the code needed to create the item labeled **Question 2**.
 - `new Friend[3];`
 - `new Friend*[3];`
 - `new *Friend[3];`
 - `Friend* MyFriends[3];`
 - none of the above.
- What is stored at the memory location labeled **Question 3**?
 - Friend** object.
 - 2101
 - 2108
 - 2112
 - none of the above.

- A pointer variable:
 - Is used to store addresses of other items in memory.
 - Can be created in either *program stack* memory or *heap* memory.
 - Can be used to release memory previously allocated using **new**.
 - Has a size of exactly 4 bytes.
 - All of the above.

For the following 5 questions, assume that the following class specification is part of the program:

```
class Actor {
private:
    char szName[12];
    int nStrength;
    int nHP;
public:
    // many public functions
}; // end class Actor
```

- Which statement will create a dynamic array of 100 **Actor** objects:
 - `Actor** p = new Actor*[100];`
 - `Actor* p = new Actor[100];`
 - `Actor aaActors[100];`
 - `Actor* p = new Actor*[100];`

- e) none of the above.
6. Which statement will create a dynamic array of 100 pointers to **Actor** objects:
- a) **Actor** p = new Actor*[100];**
 - b) **Actor* p = new Actor[100];**
 - c) **Actor aaActors[100];**
 - d) **Actor* p = new Actor*[100];**
 - e) none of the above.
7. In question #6, where will items be allocated:
- a) **p** : *program stack* **new Actor** : *program stack*
 - b) **p** : *program stack* **new Actor** : *heap*
 - c) **p** : *heap* **new Actor** : *program stack*
 - d) **p** : *heap* **new Actor** : *heap*
 - e) **p** : *no variable (item b)* **aaActors** : *heap*
8. What will be the size (in bytes) of the array created in question #5?
- a) 4 b) 400 c) 2000 d) 2020 e) none of the above.
9. Change the string variable declaration in the **Actor** class so that it can handle a pointer to a dynamically allocated array of **char**.
- a) **char pszName[];**
 - b) **char* pszName = NULL;**
 - c) **char* pszName;**
 - d) **char pszName*;**
 - e) **char pszName* = NULL;**
10. Identify the code needed to create a dynamically allocated array of **char** using the pointer variable **pszName** (This code will reside in one of the **Actor** class functions.)
- a) **pszName = new char[nStrLen + 1];**
 - b) **pszName* = new char[nStrLen + 1];**
 - c) **pszName = new char*[nStrLen + 1];**
 - d) **pszName* = new char*[nStrLen + 1];**
 - e) none of the above.
11. The memory associated with **pszName** can be released with the following statement:
- a) **delete pszName[];**
 - b) **delete *pszName;**
 - c) **delete [] pszName;**
 - d) **free pszName;**
 - e) **delete pszName;**

For the following question, assume that the following class specification is part of the program:

```
class Army {  
private:  
    // implement pointer to array of Actors  
    int nNumActors;  
public:  
    // many public functions  
}; // end class Army
```

12. Identify the code needed to declare a pointer to an array of dynamically allocated **Actor** objects.
- a) **Army = new Actor[nNumActors];**
 - b) **Actor* = new Actor[nNumActors];**
 - c) **Army* Actor*;**
 - d) **Actor* pActor;**
 - e) none of the above.
13. To input data from a disk file you will:
- a) include the file **ifstream**
 - b) create an object of type **ifstream**.
 - c) associate the **ifstream** object with a named file on disk.
 - d) use the **ifstream** object with the >> operator to store file data in variables.
 - e) all of the above.
14. Closing an object of type **ofstream** means:
- a) All data in the file is overwritten.
 - b) Output buffers are flushed to disk.
 - c) The **ofstream** object is deleted.
 - d) The disk file associated with the **ofstream** object is deleted.
 - e) None of the above.
15. Imagine that you have created an object in a block of code using **new** and captured its address in a pointer variable. The memory for that object released when:
- a) Program execution leaves the block where **new** was used to allocate the object.
 - b) The pointer with the object's address goes out of scope.
 - c) The next function call is made.
 - d) The garbage collector determines that the object is no longer in use.
 - e) None of the above.

Name: _____

Programming (20 marks)

Overview: Strategy

You will submit two things for evaluation:

- C++ program code to solve the stated problem
- Memory map that shows the organization of objects in memory when execution is underway.

Because you have only a limited amount of time, you will not need to create a solution that captures the range of capabilities that you might with a full lab.

To make the most effective use of your time, you should pursue problem solving systematically:

- Read the problem.
- Take a few minutes to list your assumptions based on your read of the problem. This will give you a list of things that must be done; and things that are not required in your solution (even though they might be appropriate in a fully functioning solution).
- Write a rough draft algorithm that solves the problem.

Identify how you would test your program solution.

The preceding steps are essentially the steps you would have taken had this been regular lab work. In this test environment, you won't need to polish the quality of *assumptions*, *algorithm* and *test plan*, since you won't be submitting them for evaluation. They are merely tools to help you craft the C++ code that solves the problem.

Overview: The Problem

Write a C++ program that manages an address book of information about people. There will be two classes: **Person** and **AddressBook**. The **Person** class will be used to build individual objects, each of which contains information about a single person. The **AddressBook** class will implement a dynamically allocated array of **Person** objects that can hold a user-defined number of **Person** objects.

You will exercise these classes by establishing values in a few **Person** objects in the **AddressBook** object. You will display the contents of all objects in the **AddressBook**, and then perform some calculations on those **Person** objects.

Your C++ code should compile without error.

Details: Person Class Specification

Create a *class* to specify a **Person**. It will consist of:

Data Members

- *sName*: This can hold text. Assume that text will contain no embedded blanks (which simplifies inputting).¹
- *nPhoneNumber*: The value will be a 7-digit number which is stored as an integer (don't worry about area code).
- *nNumOfChildren*: This value identifies how many children each person has.

Member Functions

- **Set()**: This function will establish all three values in a **Person** object by prompting a user and accepting keyboard

input. *nPhoneNumber* should be validated as described above².

- **Display()**: This function outputs the three data members.
- **Display(int nMatchingPhoneNumber)**: This function outputs the three data members selectively. It will output the data only if the *int* sent as an argument matches the phone number stored in the object.

Details: AddressBook Class Specification

Create a *class* to specify the **AddressBook**. It will consist of:

Data Members

- *pointer to a dynamically allocated array of Person objects*: The size of the array will be
- *nNumItems*: This variable tracks how many of the **Person** elements in the array are actually in use.

Member Functions

- *constructor*: The constructor will establish suitable initial values. It will receive an *int* value that will be used to determine the size of the dynamically allocated array..
- **AddPerson()**: When this function is called, it will establish the values for a **Person** object by calling the **Set()** function in the **Person** class. This function will be declared and defined inside the class.
- **DisplayAll()**: When called, this function will sweep through the array of **Person** objects, calling one of the two versions of the **Display()** function for each active **Person** object. This function will be declared and defined inside the class. In the function, before searching through the **Person** objects, ask the user if they want to filter the output based on a phone number. Call the correct version of the **Display()** function.

Note: Your classes must leave data members *private*, and provide *public* functions to control access.

Details: Activities in main()

Your *main()* function will be designed to test the functionality of these two components: class **Person** and class **AddressBook**. Perform the following actions to validate the correctness of these two classes.

- Prompt the user for the maximum size of the address book; capture that input; use that size information when you create the **AddressBook** object.
- Create an **AddressBook** object.
- Call the **AddPerson()** function for your **AddressBook** object 4 times to establish 4 sets of values for 4 persons.
- Call the **DisplayAll()** function.

Memory Map (2 Marks)

Imagine that a breakpoint is set at the last line of code in *main()*. When program execution reaches this point, show the organization of the objects with all data members labeled; include sample addresses. Since the array can be rather large, use some technique to show the range of the index positions in the array, including at least the first four index positions and the last index position. (Show the index number as well.)

¹ You can use either the string data type or an array of char to hold the description. The choice is yours.

² The other data members must be input, but no validation is required because of time constraints.