

CS 12 • Winter 2003
In-lab Programming Exercise 3
Structures and Dynamic Allocation

wagner@cs.ucr.edu
Department of Computer Science and Engineering
University of California, Riverside

February 3, 2003

In this third assignment, you'll be exercising your debugging skills. The task is very simple, though not necessarily easy. You're given the full source code for a simple application I wrote and also a sample of the proper output it produces. However, I've changed a few things in the source code and you'll find that the source file you're getting will not compile.

Your first task is to find, explain, and fix all compile errors.

Next, you'll find out that the application runs incorrectly: there are bugs lurking in between the lines. Your second task will then be to find and fix those bugs. I'll give you a hint: every function has at least one error (syntactic or logical).

Note that the source code has no comments whatsoever. You'll need to try and understand what the code does without the benefit of comments and without the TA's immediate help. Needless to say, you shouldn't ask your friends either.

It's important that you try only one idea at a time and that you document your efforts as you go along because the TAs will come to see if you understand what the program is doing or what it's supposed to be doing. Also, your documented attempts to fix this program is the stuff that you should write in your explanation section.

Good luck!

List of corrections to the source code

1. There was a missing ';' at the end of the Student struct definition. This is a compiler error.

```
struct Student
{
    std::string name;
    int id;
}; // this semi-colon was missing
```

2. In the enumeration below, OPTION_CANCEL should be defined as 99, not as 990. This is not a compiler error, but it will make the program not work correctly when you try to cancel a menu request.

```
enum { OPTION_CANCEL = 990, // should be 99, not 990
        OPTION_QUIT };
```

3. In the line `student_array = new Student[rand() % 4];`, the array is created dynamically with a *random* size between 0 and 4. So, although not a compiler error, it will generate an error when running, but only sometimes! If you happen to want to enter a number of students smaller than the random number chosen, there will be no errors. The correct line should have been `student_array = new Student[num_students];`.

4. The OPTION_QUIT sections are missing the actual call to the Quit() function. Again, not a compiler error, but without this call you cannot quit by choosing the quit option from the menu.

```
case OPTION_QUIT:
{
    done = true;
    Quit(); // this line was missing
}
break;
```

5. In the `InputOneStudent()` function, the `next_student_index` variable was incremented twice, when it should have been incremented only once. This would not be an easy bug to discover just by looking at the code.

```
void InputOneStudent ()
{
    std::string name;
    int id;

    std::cout << "Name:_";
    std::cin >> name;
    std::cout << "ID_number:_";
    std::cin >> id;

    Student student = { name, id };
    student_array [next_student_index++] = student;

    std::cout << "Current_number_of_students:_"  

               << next_student_index++ // this '++' does not belong here!  

               << std::endl << std::endl;
}
```

6. In the `InputAllStudents()` function, the condition for terminating the loop is incorrect. As a result, all but the last student would have been added to the array. The correct version counts up to and including when `i` equals `num_students`.

```
void InputAllStudents ()
{
    for (unsigned int i = next_student_index;  

         i <= num_students; // the buggy version had only '<'  

         i++)  

        InputOneStudent ();
}
```

7. Similarly, in `PrintAllStudents()`, the loop starts at 1 when it should start at 0, which causes the first student record to be missed.

```
void PrintAllStudents ()
{
    for (unsigned int i = 1; // should be 0, not 1
         i < next_student_index; i++)
    {
        PrintOneStudent(student_array[i]);
    }
}
```

8. In the function `SearchById(Student &student)`, the return statement always searches for a student with an ID equal to zero.

```
bool SearchById(Student &student)
{
    std::cout <<
        "Please enter the ID of the student to search for: ";
    int id;
    std::cin >> id;

    return Search(0, student); // should be 'return Search(id, student);'
}
```

9. `bool SearchByName(Student student)` should be `bool SearchByName(Student &student)` or even `bool SearchByName(const Student &student)`, since we typically do not want to pass structures by value and, in this case, the structure isn't changed so we might as well have the `const` there too for good measure.

10. I had `bool Search(const int student_id, Student &student)` always return `false` because I wanted **you** to implement the search function. I didn't expect anyone to implement a binary search, so here's the linear search version:

```
bool Search(const int student_id, Student &student)
{
    bool found = false;

    for (unsigned int i = 0; i < next_student_index && !found; i++)
    {
        Student student_in_array = student_array[i];
        if (student_in_array.id == student_id)

```

```

        {
            student = student_in_array;
            found = true;
        }
    }

    return found;
}

```

11. I also had `bool Search(const std::string student_name, Student &student)` always return `false`. Here's the linear search version:

```

bool Search(const std::string student_name, Student &student)
{
    bool found = false;

    for (unsigned int i = 0; i < next_student_index && !found; i++)
    {
        Student student_in_array = student_array[i];
        if (student_in_array.name == student_name)
        {
            student = student_in_array;
            found = true;
        }
    }

    return found;
}

```

12. Another use of a random number, to throw you off a bit... The correct version should output `student.id` instead of `(rand() % 23)`.

```

void PrintOneStudent(const Student &student)
{
    std::cout << "Name:_ " << student.name << std::endl;
    std::cout << "ID_#:_ " << (rand() \% 23) << std::endl << std::endl;
}

```

13. In the function `Quit()`, `cout` is missing the namespace identification. Also, there's a missing `<<` before the `std::endl`. Both would cause the compiler to complain.

```

void Quit()
{
    cout << // this should have been 'std::cout <<'
    "Thank_you_for_using_the_CS12_Student_Information_Center." <<
    "Goodbye_now. :)" << endl << endl;
    // should be "Good bye now. :)" << std::endl << std::endl;

    exit(0);
}

```

14. Finally, the last correction is another `std` (actually, two) missing. The correct version should have `std::endl` insted of `endl`:

```

void PrintError(std::string error_message)
{
    std::cout << "ERROR:_" << error_message
               << endl << endl; // missing 'std::' before both endl's
}

```

■