Exercises 1.5

9. What are some of the ways in which student programs differ from real-world software?

Solution:

Answers will vary.
Answers may include:

- Size of the program
- Student programs are standalone (not part of a system)
- Complexity of the problem being solved.
- Errors may have more severe consequences in real-world software.
- Lifetime of student programs is often shorter than real-world software.

10. Name three kinds of programming errors and give examples of each. When during program development is each likely to be detected?

Solution:

- *Syntax errors*: violations of the grammar rules of the high-level language in which the program is written.
  - Example: `int t` would be a syntax error in C++ since there is no semi-colon at the end of the statement. The correct statement would be: `int t;`
  - Syntax errors are likely to be detected during program compilation.
- *Run-time errors*: Errors that occur during program execution.
  - Example:

```
int size=10;
for (int i=0; i<size; ++i) {
   int div = size/i;
}
```

    This code will produce a run-time error since it will try to divide by zero.
  - Run-time errors are likely to be detected during program execution.
- *Logic errors*: Errors in some aspect of the design - most often an algorithm - on which the program unit is based.
  - Example:

```
int sum(int a, int b) {
   return a-b;
}
```

    This code has a logic error since the function should add two integers, but returns the difference between the two integers.
  - Logic errors are likely to be detected during program execution.

12. Find some other examples of "software horror stories" and write brief reports for each, describing the error and what harm or adversity resulted from it.

Solution:

Answers will vary.

Chapter 1 Programming Problems

4. The following function performs a linear search of a list l of length ll for the item it, returning 0 or 1 depending on whether it is not found. Many principles of good programming are violated. Describe some of these and rewrite the function in an acceptable format.

```
int LS(int l[], int ll, int it)
/* Search l for it */
{
int i=0, f=0; A:if (l[i]==it)
goto B; if (i=ll) goto
C;/*ADD 1 to i*/i++;goto A;
B:f=1;C:return f;
}
```

Solution:

Principles that are violated:

- Multiple statements on one line of code
- Single statements are split into multiple lines of code
- No indention is used for blocks of code.
- The code uses "goto" commands. We should use a for look to iterate.
- Comments for the function should include preconditions and post conditions as well as the purpose.
- Variable names are not descriptive
- The function name is not descriptive
- The function uses global variables
- The input parameters are not expected to change so they should be declared constants. They should also be passed by reference.
- The function should be boolean since it is determining whether or not something is in the input array.

The rewritten function:

```
/**
 * Search an array of integers to determine if a specific item is present
 *
 * Preconditions: list is a list of integer values.
 *      size is the length of the list - size must be a non-negative integer.
 *      item is the item we are searching for.
 * Postconditions: return false if item is not found. return true if item is found.
 *
 * Worst-Case Time Complexity: O(n)
 */
boolean isItemInList(const int list[], const int& length, const int& item) {
   boolean isFound = false;

   for (int i=0; i<length; ++i) {
      if (list[i]==item) {
         isFound = true;
         break;
      }
   }

   return isFound;
}
```

Exercises 2.4

16. Describe the output produced by the following statements

```
int * foo, * goo;

foo = new int;
*foo = 1;
cout << (*foo) << endl; // output: 1
goo = new int;
*goo = 3;
cout << (*foo) << (*goo) << endl; // output: 13
*foo=*goo+3;
cout << (*foo) << (*goo) << endl; // output: 43
foo = goo;
*goo = 5;
cout << (*foo) << (*goo) << endl; // output: 55
*foo = 7;
cout << (*foo) << (*goo) << endl; // output: 77
goo = foo;
*foo = 9;
cout << (*foo) << (*goo) << endl; // output: 99
```

Solution:

The output for the above code is:

```
1
13
43
55
77
99
```