

1. chapter 1 – Many compilers have abandoned the *void main()*. It may be necessary to go back to the older minimal *main* function outline:

```
int main()
{
    return 0;
}
```

2. chapter 1, page 10 – The pause code is outdated. Two alternatives were omitted. The first alternative works only on a Microsoft operating system that supports DOS commands. It uses the *system* function from the *cstdlib* library to issue a DOS *pause* command.

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    cout << "15 times 3 equals " << 15 * 3;
    system("PAUSE");
    return 0;
}
```

The second example uses the *get()* method from the *iostream* library. It reads in a character, including spaces and the end-of-line.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "15 times 3 equals " << 15 * 3;
    cout << "\n... Press Enter to Continue ...";
    cin.get();
    return 0;
}
```

3. chapter 6, page 11 – Discussion of the problems of switching from the input stream extraction operator (*>>*) to *get* and both forms of *getline* was omitted. It is as follows:

The input stream extraction operator (*>>*) stops at a space or end-of-line. If the next input is with the input stream extractor, the space or end-of-line is skipped. However, if the next input after a input stream extractor is a *get* or *getline* (either form), the space or end-of-line will satisfy the *get* or *getline* and it may appear that no input has taken place and that the program did not allow the input, from the keyboard or a file. This can be remedied in various ways. The simplest is to put a “dummy” read between the use of the input stream extractor and the *get* or *getline*. For example:

```

#include <iostream>
#include <string>
using namespace std;

int main()
{
    int a;
    string dummy, s;

    cout << "Enter an integer: ";
    cin >> a;
    cout << "Enter a word: ";
    getline(cin, dummy);
    getline(cin, s);
    cout << "The integer is: " << a << endl
         << "The word is:      " << s << endl;

    return 0;
}

```

Here is an example of a run of the above code:

```

Enter an integer: 3
Enter a word: fred
The integer is: 3
The word is:      fred

```

4. chapter 11, page 5 – programming assignment 11.1 should read:

Put the volume functions of programming assignment 10.1 in a header file. Include the header file in a program that uses the functions found in that header file.

5. chapter 12, page 9 – “#define _CARD” should be “#define _CAR”
6. chapter 12, page 13
- “#include <conio.h>” should be removed; “getch()” should be changed to “cin.get()”
 - “Press any key to continue” should be changed to “Press Enter to continue”
 -
7. chapter 12, page 14 – not an error, but here is an alternative representation of class “student”

```

class student {
public:
    student();                // default constructor creates student noone

    // preferred constructor
    student(string initName, string initId, int initYearInSchool);

    string returnName();     // returns the name of the student

```

```

        string returnId();           // returns the Id of the student
        int returnYearInSchool();    // returns the grade (level) of the student
        void promote();             // adds one to the grade (level) of the student
    private:
        string name;
        string id;
        int yearInSchool;
};

```

8. chapter 13, page 1 – in the class student, the default constructor definition is badly formed; it should read “student();”
9. chapter 13, page 2 – in the example program, the declaration of student2 needs to end in a semi-colon (;)
10. chapter 13, page 7 – in methods setGrade, getStudent and getGrade, the variable “x” should be declared outside of the for loop in order to be available after the for loop (old fashion code to declare the variable inside the for loop and then use it after the for loop); example:

```

int x;
for (x=0; ...

```

11. chapter 19, page 3 – “#include <conio.h>” should be removed, “beeps = getch();” should be changed to “beeps = cin.get();”
12. chapter 20, page 11, exercise 20.15 – library header files <time> and <stdlib> should be written <ctime> and <cstdlib> in order to use the newer versions available when using a namespace
13. chapter 22, page 3, exercise 22.1 – remove “Use a *getch* function as part of this addition to the program.”
14. chapter 22, page 3, exercise 22.2 – should read :

This program will use the class Car from chapter 12.

Write a program that accepts two characters the from user until the user decides to halt. Each character represents a vehicle of class Car. Possible vehicle symbols are ‘A’, ‘B’, ‘C’, ‘D’ and ‘E’. Each symbol is associated with a counter.

Each time two characters are read in, they are to become the symbols of two objects of class Car. The two Car objects then race each other by being moved 10 times each. Based on their locations, a winner or a tie is determined. If a Car object wins the race, its counter (based on the current symbol of the Car object) is incremented by +2. If there is a tie, the counters of both Car objects are incremented by +1.

The results of each race are to be output to the monitor screen. In the cases where there is a winner, this will include the name of the winner and the distance the winner has gone. In the case were there is a tie, the output will consist of the name of both contestants and the distance that they have traveled.

When the user decides to quit, the program should output to the monitor screen the symbol and score of each vehicle. For example:

A – 59
B – 37
C – 63
D – 27
E – 84

15. chapter 25, exercise 25.6 – add to the “(Note:)” at the end of the first paragraph:

“or set the minimum or maximum to the value of the first element in the array”

16. chapter 25, exercise 25.7b – add at the end :

“Have the program output the number of elements in the array used.”

17. *more to come ...*