Problem 1. If \( a_1 \) is true and \( a_2 \) is false, determine whether the following returns a true or false.

(a) \( a_1 \lor \neg a_2 \)

(b) \( a_1 \land \frac{7}{3} > 2 \)

(c) \( a_1 \land 82 \% 5 < 7 \)

(d) \( \neg a_1 \lor a_2 \)

(e) \( \neg a_1 \land \neg a_2 \)

(a) True, (b) False, (c) True, (d) False, (e) False

Since \( a_1 \) is true and \( a_2 \) is false, \( \neg a_1 \) is false and \( \neg a_2 \) is true, so that

(a), true \( \lor \) false, which returns a true.

(b), integer division uses truncation, so that \( \frac{7}{3} \) is 2, and \( 2 > 2 \) returns a false. The final answers is true \( \land \) false, which returns a false.

(c), \( 82 \% 5 \) is 2, so that the final answer is true \( \land \) true, which returns a true.

(d), false \( \lor \) false returns a false.

(e), false \( \land \) true returns a false.
Problem 2. Give the output of the following program:

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int i(1), j(2), k(3);

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);

    cout << ++i << " , " << j++ << endl;
    cout << i << " , " << j << endl;

    k += i -= j;
    cout << i << " , " << j << " , " << k << endl;
    cout << k + 3.6 << endl;
    cout << (double)k + 3.6 << endl;

    return 0;
}
```

2, 2  
2, 3  
-1, 3, 2  
5.6  
5.6

- The line 1 of output: \(i\) is increased by 1 before being printed, so a 2 is printed for \(i\). The variable \(j\) is increased by 1 after being printed.

- Line 2: now \(i\) is 2 and \(j\) is 3.

- Line 3: The assignment goes from right to left: Subtract \(j\) from \(i\) first. Then add \(i\) to \(k\). The values of \(i\), \(j\), \(k\) are then printed.

- Line 4: int + double gives a double. Line 4 should be the same as line 5.

- `cout.precision(1)` has no impact on integers. Changing the first line to `2.0, 2.0, etc` is an error.
Problem 3. Give the output of the following program:

```cpp
#include <iostream>
using namespace std;
int main()
{
    for (int k = 1; k < 18; k++)
    {
        if ( k % 5 == 0 )
            cout << " k is equal to " << k << endl;
        else
            if ( k % 7 == 0 )
                cout << k*10 << endl;
    }

    return 0;
}
```

```
- Between 1 and 18, k is divisible by 5 when k is 5, 10, 15, respectively. So the corresponding statements are printed. Note that there is a space before the first character.

- The only k divisible by 7 in this range is 7 and 14, so that the corresponding statement are printed. Note that there is no space before the first letter.
```
Problem 4. Write the program below using a while-loop instead of the for-loop

```c
for( i = 11; i>=4;i--) {
    some code ...
}
```

```
i=11;
while(i>=4) {
    some code ...
    i--; // or --i; or i=i-1;
}
```

It is important that you not decrement \( i \) until after you have executed the code since the code may use the value of \( i \). It is also important that you not write `int i(11);` or some such. The variable was not declared in the for-loop so it must have been declared higher up in part of the code that you are not being shown. Hence `int i(11);` is not right. IF the for-loop had started `for( int i=11; i>=4;i--)` then you need to write

```c
{int i(11);
while(i>=4) {
    some code ...
    i--; // or --i; or i=i-1;
}
}
```

You might wish to wonder why the extra pair of braces are necessary.
Problem 5. Write a recursive function with declaration

long F(short);

which returns 1 if the input is not positive and the product of the first $n$ odd numbers if $n$ is positive.

long F(short s)
{
if(s<1) {return(1);} // or: if(s<=0) {return(1);} 
else {
    return( (2 * s - 1) * F(s-1)); 
}
}

- The most common mistake was to solve the problem without using recursion.
- The next most common problem was to output something other than the product of the first $n$ odd numbers.
- You do not need to name the variable in the declaration long F(short); but you must name it when you write the code. You can call it something other than s but you must call it something.
Problem 6. Give the output of the following program:

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

int modify(int , int &);

int main()
{
    int j = 3, m = 8;
    modify(j,m);
    j = modify(j,m);
    modify(j,m);
    return 0;
}

int modify(int j, int & m)
{
    m += 7;
    j *= 2;
    cout << "m is now " << m << endl;
    cout << "j is now " << j << endl;
    return j;
}
```

m is now 15
j is now 6
m is now 22
j is now 6
m is now 29
j is now 12

- The first variable j is used by the function modify “passing by value”, the second variable m is used by the function modify “passing by reference”.

- The first modify function use 3 for the first variable j, and use 8 for the second variable m. After this call, the variable j remains 3 in the main function, while the variable m is changed to 15.

- The second modify function use 3 for the first variable j, and use 15 for the second variable m. After this call, the variable j was reassigned by the number returned by the function modify (which is 12), while the variable m is changed to 22 by passing by reference.

- The third modify function use 6 for the first variable j, and use 22 for the second variable m.
Problem 7. Describe the Monte Carlo Method for computing the following integral

\[ \int_0^1 x \sin \frac{1}{x} dx. \]
```cpp
#include <iostream>
#include <cmath>  // for the sin(x) function
#include <ctime>  // for the time() function
#include <cstdlib> // for the rand() function
using namespace std;

int main()
{
    double x, y, prob, area;
    int hit = 0, total = 1000000;

    srand(time(NULL));
    //generating the seed value for the random numbers

    for (int k = 1; k <= total; k++) {
        x = (double)rand()/RAND_MAX; y = 2*(double)rand()/RAND_MAX;
        //generates a random x between 0 and 1, and y between 0 and 2
        if ( y <= x*sin(1/x) + 1 ) hit++; //if under the curve, then a hit
    }

    prob = (double)hit / (double)total;
    area = prob*2.0;  // The Box has area 2
    area = area - 1; // subtract 1 to obtain the original area
    cout << "The estimated value of the integral is: " << area << endl;
    return 0;
}
```

- For an area $A$ in a box with area $B$, the probability of hitting the area $A$ when throwing a dart into the box is given by $\frac{\text{area}(A)}{\text{area}(B)}$. Using random generator, we can simulate this process so that

$$\frac{\text{area}(A)}{\text{area}(B)} = \frac{\text{number of hits to area}(A)}{\text{total simulations}}.$$ 

from which we can compute the approximate value the area of $A$.

- Since $x \sin \frac{1}{x}$ changes sign, the integral is not directly given by an area. One can first compute

$$\int_0^1 \left(x \sin \frac{1}{x} + 1\right) dx$$

as an area above the $x$-axis and then subtract 1 from it to obtain the desired integral. Alternatively, one can compute the hits above $x$-axis under the curve $y = x \sin \frac{1}{x}$, and subtract the hits below $x$-axis above the curve $y = x \sin \frac{1}{x}$. 
Problem 8. Consider the following bit of code:

```c
xx=2;
switch(cc) {
  case 'a': xx+=3;
  case 'b': xx %= 3; break;
  case 'c': xx-=4;
  default: xx=0;
}
```

What value will xx have if cc='a'? How about if cc='c'?

If cc = 'a' then you do case 'a' so x=5 and then you drop into the code for case 'b' which computes the remainder of the current value of x - namely 5 - divided by 3 so at the end of the day, x=2.

If cc='c' then you do the code for case 'c' which sets x to -2 and then you drop into the default code so you set x=0. At the end of this bit of code x=0.

The most common mistake was to behave as if each case statement ended with a break; - only one of them does!
Problem 9. Consider the following bit of code

```c
if( test_1 ) { code_1 }
else if( test_2 ) { code_2 }
else if( test_3 ) { code_3 }
else { code_4 }
```

code_after

If test_1 and test_3 are false, discuss the two ways the program can behave at this point (i.e., indicate which codes are executed, and which codes are not executed).

We are told test_1 and test_3 are false: the two ways the program can behave at this point are the result of test_2 which can be either true or false.

Since test_1 is false, code_1 is never evaluated. If test_2 is true, then code_2 is evaluated and then the program jumps to code_after.

If test_2 is false, then code_2 is not evaluated. Since test_3 is false, code_3 is not evaluated either. Finally, code_4 is evaluated, followed by code_after.

In summary:

- test_2 true: we do code_2 followed by code_after.
- test_2 false: we do code_4 followed by code_after.
This program computes the certain average of a sequence of inputs according to the formula

\[ \text{average} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} x_i^2}. \]

Any number ending at 0 will not be counted. The player will win 100 points or 200 points if this number turns out to be 100 or 200; otherwise it print out a message to try again.

Problem 10. There are many errors in this code. Please indicate or circle all errors.

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

int main()
{
    double x, sum;
    int count;

cout << "enter a positive number : ";
cin >> x;

while ( !cin.eof() )
{
    if ( x%10 == 0 )
        x = 0;
    else
    {
        sum += x^2;
        count++;
    }

cout << "enter a positive number : ";
cin >> x;
}

switch( sqrt(sum/count) )
{
    case 100:
        cout << "You win 100 points" << endl;
    case 200:
        cout << "You win 200 points" << endl;
    default:
        cout << "Try again later" << endl;
}
return 0;
}
```
Here is the correction:

```cpp
#include <iostream>
#include <cmath> // it is necessary since sqrt is called
using namespace std;

int main()
{
    double x, sum = 0.0; // need to initialize
    int count = 0; // need to initialize

    cout << "enter a positive number : ";
    cin >> x;
    while ( !cin.eof() ) // This is OK - eventually type ^D to exit (see p. 108)
    {
        if ( (int)x%10 == 0 ) // % only applies to integers
            x = 0;
        else
        {
            sum += x*x; // or use pow function. ^ is not allowed
            count++;
        }
        cout << "enter a positive number : ";
        cin >> x;
    }

    int ans = sqrt(sum/count);
    // The idea is to switch on ans which is an int and hence acceptable.
    // BUT sqrt(sum/count) may not be exactly 100 or 200
    double test = sqrt(sum/count) - ans;
    if(test != 0.0) {ans=0;}
    switch( ans )
    {
        case 100:
            // You want the number 100, not the string of 3 characters '100'
            cout << "You win 100 points" << endl;
            break; // need a break here
        case 200:
            cout << "You win 200 points" << endl;
            break; // need a break here
        default:
            cout << "Try again later" << endl;
            break; // this break is optional
    }

    return 0;
}
```