1. Which of the following declarations are illegal, and why? Be sure to state your reasons.

   (a) `int double;`
   ANS: You can't use a keyword like `double` as a variable name.

   (b) `double 33.7;`
   ANS: You can't use a literal like 33.7 for a variable name.

   (c) `bool end?;`
   ANS: You can't use punctuation in a variable name.

   (d) `unsigned int next-number;`
   ANS: You can't use operations in a variable name.

2. Each of the following programs has error(s). Locate the error(s), classify each error as either a syntax error, or a logical error and then correct it.:

   (a)
```
#include <iostream>

int main()
    double x = 2.7e-1;
    int y = x;
    int a = 1000;
    char b = a;
    std::cout << "\ny = " << y << " and " << "\nb = " << b << endl;
}
```
   ANS: The syntax error is the lack of an opening curly brace for the body of `main()`. Also, there are narrowing errors when `int y = x` and when the value of `int a = 1000` is assigned to `char b`. This is a kind of logic error that will actually make the value of `b = 1000%256`, which is 232 or 'Φ'.

   (b)
```
#include <iostream>
int main() {
   float n /= 10.1;
   std::cout << n*10.1: }
```
   ANS: There are several syntax errors here. First, performing division on an uninitialized variable in `float n /= 10.1` does not parse. Second, there is a colon where there should be a semicolon on the next line.

3. Write a loop that will cause an integer overflow error in any compiler.
   There are a variety of ways to do this, including this one-liner (no terminating condition):

```
for(int i = 1;;++i);
```

4. Determine whether each boolean expression is `true` or `false`, assuming `w=2, x=3, y=5` and `z=7`.

   (a) `x-2==1 && z+x==10` is `true AND true` is `true`.

   (b) `++x == --y` is `4 == 4` which is `true`.

   (c) $w\%x * y > 10$ `||` $y\%z * x == 14$ is $2\%3 * 5 > 10$ `OR` $5\%7 * 3 == 14$ is $10 > 10$ `OR` $15 == 14$ which is `false OR false` which is `false.`

5. What is the output of the following program if the user inputs `1600`?

```
#include <iostream>
int main() {
    int murbs, hurbs, durbs, x;
    std::cout << "\nInput a an integer: ";
    std::cin >> x;
    durbs = x/(4*6);
    hurbs = x%(4*6)/3;
    murbs = x%(4*6)%6;
    std::cout << std::endl << x << " murbs is equivalent to " << durbs
        << " durbs  and " << hurbs << " murbs";
}
```

ANS: Threading through we have `durbs = 1600/24 = 200/3` will be 66 (integer division) and then `hurbs = 22` and `murbs = 0` so the output will be "`1600 murbs is equivalent to 66 durbs and 5 murbs`"–clearly misrepresenting the truth of the matter!

6. Consider the following program fragment.

```
int accumulator = 0, sam, pam;
cout << "\nEnter integers for sam and pam: ";
cin >> sam >> pam;
while (true ) {
   if (pam == 0) break ;
   accumulator += ((pam % 2 == 1) ? sam : 0);
   pam /= 2;
   sam *= 2;
}
cout << accumulator << "\n";
```

sam	pam	accumulator
5	4	0
10	2	0
20	1	0
40	0	20

sam	pam	accumulator
6	17	0
12	8	6
24	4	6
48	2	6
96	1	6
102	0	102

(a) Complete the following tables until the program completes, or indicate that it's an infinite loop.
ANS: See above.

(b) In a few words, describe the output of this program in terms of the input values of `sam` and `pam`, and how it works.
First, note that the code would be tightened up by not calling a break, but conditioning the loop like so: `while(pam!=0)`.
Essentially, `pam` is decomposed as a sum of powers of 2, and `sam` is distributed. This results in the product, `sam*pam`. For example $5 * 2^2$ and $6 * (1 + 2^4)$

7. Consider the following program:

```
#include <iostream>

int main() {
    int i = 5;
    for(int i = 1; i < 10; ++i) {
        std::cout << i*i << '\t';
        if(i%3 == 0) cout << endl;
    }
    std::cout << "\ni = " << i << std::endl;
}
```

(a) What is the output of the program? **ANS:**

```
1   4   9
16  25  36
49  64  81
i = 5
```

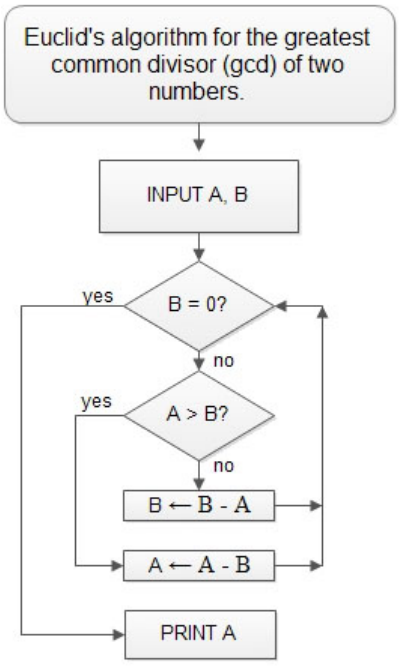
(b) Rewrite the for loop as an equivalent while loop so that output doesn't change. **ANS:**

```
int j = 1;
while(j < 10) {
   std::cout << j*j << '\t';
   if(j++%3 == 0) std::cout << std::endl;
}
```

8. The sum of the first n cubes is given by $1+2^3+3^3+\cdots+n^3=\left(\frac{n(n+1)}{2}\right)^2$. Write a complete program that checks this formula by inputting n and then computing and comparing the values of both sides of the equation.

```
int main() {
    int n, sum = 0;
    cout << "\nTo add the first n cubes, 1 + 8 + 27 + ...."
         << "\nEnter n: ";
    cin >> n;
    for(int i = n; i >= 0; --i)  {
        sum += i*i*i;
    }
    cout << "\nThe sum of the first " << n << " cubes is " << sum << endl;
    cout << "\n(n*(n+1)/2)^2 = " << n*n*(n+1)*(n+1)/4 << endl;
    return 0;
}
```

9. Consider the flow chart for a greatest common divisor algorithm shown at right.



(a) Tabulate values for `A` and `B` as the algorithm progresses. Start with `A = 1232` and `B = 1190`.

A	B
1232	1190
42	1190
42	1148
42	1106
42	1064
⋮	⋮
42	14
28	14
14	14
14	0

(b) Write a C++ function that takes integer parameters `A` and `B` and returns the output of this algorithm.

There are many ways to do this. On the left we follow the algorithm above, on the right is a more efficient, recursive method.

```
int main() {
    int A{0}, B{0};
    std::cin >> A >> B;
    while(B) {
        if(A>B) A-=B;
        else B-=A;
    }
    std::cout << A;
}
```

```
int gcd(int A, int B) {
     if(B==0) return A;
     return gcd(B,A%B);
}
int main() {
    int A{0}, B{0};
    std::cin >> A >> B;
    std::cout << gcd(A,B);
}
```