1. $3 \quad 8 - 12/5 = 8 - 2 = 6$
2. $3 \quad 14.0/3.0 + 5.0 = 4.6667 + 5.0 = 9.6667$
3. $2 \quad 3%7 - 4 = 3 - 4 = -1$
4. $1 \quad 4.2$ is double, so $4.2 - 6 = -1.8$
5. $2 \quad$ floats shown with decimal point
6. $5 \quad 2.5$ is double, so double division is used $\Rightarrow 4.8$
7. $1 \quad$ assigning double to int results in truncation of $6.9$ to $6$
8. $1 \quad$ linus * ++lucy : lucy is incremented to $0$, then the multiplication occurs
9. $1 \quad$ lucy + $5 = 4$ = logically true (integers in Boolean context are OK)
10. $2 \quad$ ++ is unary, so cannot be used as the operator between two operands
11. $1 \quad$ pigpen $\leq 1$ evaluates to false, which is the integer value $0$ in C++

For Questions 12 – 16

```c
int zero = 0, one = 1, two = 2, three = 3, four = 4;

infile >> zero >> one >> zero >> two; // reads values $43, 21, 84, 97$
infile.ignore(100, '\n'); // skips to beginning of line 2
infile >> zero >> two; // reads values $17, 32$
infile.ignore(100, '\n'); // skips to beginning of line 3
infile >> three >> three >> four; // reads values $51, 83, 13$
infile.ignore(100, '\n'); // skips to beginning of line 4
infile.ignore(100, '\n'); // skips to EOF
if (!infile.eof()) infile >> one; // false, so no read done
```

12. $1$
13. $2$
14. $2$
15. $5$
16. $3$

For Questions 17 – 19

```c
int i1;
char ch1 = 'R', ch2 = 'S';
ifile.get(ch1); // reads '4'
ifile.get(ch2); // reads '	'
ifile >> i1; // reads 25
ifile.get(ch1); // reads '	'
ifile.get(ch2); // reads '3'
ifile >> i1; // reads 7
```

17. $5$
18. $4$
19. $10$
For Questions 20 and 21
The program reads three numbers at a time, and computes their average and the total of all the numbers in the program. The output is

6
2
7
46

20. 1
21. 4

For Questions 22 and 23
Only the extraction operator is used, which ignores white space.

```c++
ifile >> i1 >> ch1 >> ch2 >> ch3;
// reads 4, '2' '5' '3'
```

22. 3
23. 10

For Questions 24 – 28

```c++
int anInt1, anInt2;
float aFloat1, aFloat2, aFloat3;
rosebud >> aFloat1 >> anInt1;  // reads 17 (converts to float), 72
cout << aFloat1 << " " << anInt1 << " ";
rosebud.ignore(4, '\n');  // skips to the 8 of -8.3 in first line
rosebud >> aFloat2;  // reads 8.3
rosebud.ignore(80, '\n');  // skips to beginning of line 2
rosebud >> anInt2 >> aFloat3;  // reads 4, .2
cout << aFloat2 << " " << anInt2 << " " << aFloat3;
```

24. 6
25. 6
26. 6
27. 10
28. 3

29. 1 !bunny || !cat = true || false = true
30. 3 <> is not a valid C++ operator
31. 2 (a + 2 * b) / 3 = 2; 2 && 0 = false
32. 1 (b * b - 4 * a * c) = 1 and 1 > 0 = true; 2 * a * b = 8, and 8 > 0 = true, and true && true = true
33. 3 => is not a valid C++ operator
34. 2 (S + T % R >= Q + R) is true, so go into first if clause and Q becomes 2 
    (T * 3 / S != R) is true, so go into second if clause, and Q becomes 1
35. 2 \((\text{golfScore} < \text{par})\) is true, so go into first if clause
\((\text{golfScore} \leq \text{par} - 5)\) is false, so skip the if clause
IMPORTANT: the else is positioned poorly, but IS associated with the
nearest unmatched if, so the else clause is executed now
The output is printed without double quotes

36. 5 \((x \% 2 = 1)\) is true, so gamma becomes 5
\(x\) becomes 4
\((x / 2 == 0)\) is false \((4/2 = 2)\), so skip statement in if
\(x\) becomes 5
\((x \% 2 = 1)\) is true, so gamma becomes 10

37. 7 The logical negation of \((Q1 >= Q2)\) is \((Q1 < Q2)\) thus 1 is equivalent since
it uses the negation, and switches the associated operations. 3 is equivalent,
since it expresses the if-else as two if statements. 2 and 4 are not equivalent,
since if \(Q1 == Q2\), different outcomes are produced than in the original

38. 3 Goes to case 2, setting middle to ‘B’, but since there is no break, continues
to next statement and sets middle to ‘C’

39. 4 Goes to case 4, setting middle to ‘D’; continues to default statement, but
this does not affect middle

40. 5 Default is executed, which does not change middle, so it retains its
original value of ‘#’