HW #4

Pdf file only, with heading:
HW #4
Mitchell, Crane
2, A55587424
Due: 11:00 pm, Sunday Sept. 30, 2018 at Google Classroom

1. (5 pts) Write an expression that evaluates to 1 if a is the smallest among integers a, b, and c and to 0 otherwise. If you used any parentheses, you must tell whether each pair of parentheses is redundant.
   a < b && a < c

2. (5 pts) Write an expression that evaluates to 0 if x is between 10 and 100 and 1 otherwise.
   !(x > 10 && x < 100)

3. (5 pts) Give the output of the following program fragment. Explain how alpha is computed to the value you give.

```
int alpha = 10, beta = 5, gamma = 3;
alpha /= beta++ - --gamma;
printf("%d %d %d\n", alpha, beta, gamma);
```

   3 6 2
   beta is subtracted by gamma after gamma has been decremented. Then beta is incremented The right side of the expression now equals 3 (5 – 2). alpha is divided by this result the result is stored in alpha. The result is 3 (10 / 3 = 3)

4. (7 pts) Give the values of x and y after the following statements are executed. Provide a fully detailed explanation about how you arrive at such values, including all the binary forms.

```
int x = 18;
x = x >> 2;
int y = 12;
y = y << 4;
```

   x = 4; y = 192
   x = 18 = 10010

x >> 2 means the bits are shifted to the right two spaces. X now equals 00100 or 2^2 or 4. y = 12 = 1100. y << 4 means the bits are shifted to the left four spaces. Y now equals 11000000 or 2^7 + 2^8 or 128 + 64 or 192.

5. (7 pts) Give the value of variable named result after the two statements below are executed. Provide a fully detailed explanation about how you arrive at this value, including all intermediate terms to be summed.

```
int x = 5, y = 7 ;
int result = (x == y) + y/2 + !y;
```

The value of result is 3. x == y evaluates to 0 because x does not equal y. 0 is added to y/2 which evaluates to 3 because integer division has an integer result. 3 is now added to not y. since y is a non-zero integer, !y evaluates to 0, so result = 0 + 3 + 0 or 3.

6. (8 pts) Give the output from the following program fragment. Provide a fully detailed explanation about how you reach such an output, including all the binary forms.

```
int x = 17, y = 4;
int z = x & y;
int w = x && y;
printf ("z = %d\tw=%d", z, w);
```

z = 0 w=1
z = x & y evaluates to 0 because:

x = 17 = 10001
y = 04 = 00100

x & y = 00000

w = x && y evaluates to 1 because x and y are both non-zero integers

7. (8 pts) Write an if statement that outputs one of the letter grade letters A, B, C, D, F, given the numeric grade. Assume the numeric grade is stored in an integer variable named grade. Use the following scale for conversion:
   >= 90: A, 80 – 89: B, 70 – 79: C, 60 – 69: D, < 60: F

```
if (grade >= 90)
    printf("%c", 'A');
else if (grade >= 80 && grade < 90)
    printf("%c", 'B');
else if (grade >= 70 && grade < 80)
    printf("%c", 'C');
else if (grade >= 60 && grade < 70)
    printf("%c", 'D');
else if (grade < 60)
    printf("%c", 'F');
```

8. (7 pts) Give the output of the following program fragment. Provide your fully detailed explanation.

```
int x = 23;
if (24 > x > 18) {
     printf("%d is between 18 and 24\n");
} else {
     printf("%d is outside the range 18 - 24\n");
}
```

The output of this code is not proper as there is no data arguments in the printf statements. I assume that x was supposed to be an argument. If x was an argument the output would be

23 is between 18 and 24

9. (8 pts) Give the value of x after executing the following statements. Provide a fully detailed explanation for your value.

```
int a = 4, b = 2, c = 4;
x = a && b ? a/b : a/c;
```

the value of x is 2. The expression is evaluated like an if else statement. First a && b is evaluated which results in 1 because a and b are both non-zero integers. Since a && b is 1, a /b is evaluated and stored as x. a/b evaluates to 2 because 4/2 = 2.

10.(10 pts) First write the following fragment in proper indentation. Then, for each of the cases a) and b), explain in full detail about the execution of the following selection statements, and finally give the output from the fragment.

```
if (w < 10)
if (z > 10)
printf("0000\n");
else
printf("1111\n");
printf("2222\n");
```

when

a) w = 15, z = 5;
b) w = 5, z = 15;

Since it was not clear, I assumed that the last line of the code "printf("2222\n"); was outside the initial if statement. the result of a.) above is
2222
because w = 15 is not less than 10, so 2222 is output. The result of b.) above is
0000
2222
because w = 5 is less than 10 and z = 15 is greater than 10.

11.(10 pts) Convert the following if statement into a switch statement.

```
if (answer == 'y') {
      printf("Let's do it\n");
      printf("Be ready at 5:00pm\n");
      countYes++;
} else if (answer == 'n') {
```

```
 printf(“May be next time\n”);
 countNo++;
 } else if (answer == ‘u’) {
      printf(“Still undecided?\n”);
      } else {
      printf(“This is not a valid options\n”);
      }
```

```
switch (answer) {
    case 'y':
        printf("Let's do it\n");
        printf("Be ready at 5:00pm\n");
        countYes++;
        break;
    case 'n':
        printf("Maybe next time\n");
        countNo++;
        break;
    case 'u':
        printf("Still undecided?\n");
        break;
    default:
        printf("This is not a valid option\n");
        break;
}
```

12.(10 pts) Give the output of the following fragment. Provide a fully detailed explanation.

```
int delta = 26;
switch (delta/5) {
      case 1: printf("delta is small");
      case 5: printf("delta is midsize");
      case 10: printf("delta is large");
      default: printf("delta can be anything");
}
```

the output of the fragment is:
delta is midsizedelta is largedelta can be anything
because there are no break statements, the execution falls through every statement after case 5: because 26 / 5 is 5.

13.(10 pts) Convert the following **for** loop into a **while** loop.

```
for (row = 50; row >= 1; row--)
    if (row % 2 == 0)
        printf("Even iteration #%d\n", row);
    else
        printf("Odd iteration #%d\n", row);
```

```
int row = 50;
while (row >=1) {
   if (row % 2 == 0)
      printf("Even iteration #%d\n", row);
   else
      printf("Odd iteration #%d\n", row);
   row -= 1;
}
```