Name: _____

1. (10 points) For the following, Check T if the statement is true, or F if the statement is false.

   (a) [X] T [ ] F : The **char** data type in C may refer to either the ASCII code for a single letter, or a number between -128 and +127, or a list of 8 bits.

   (b) [ ] T [X] F : The left hand side of an assignment statement can be a literal constant.
   The target of an assignment must be a modifiable variable.

   (c) [X] T [ ] F : If you include stdbool.h, then you may use the extra data type of `bool` to declare a boolean variable, along with the literals `true` and `false`.

   (d) [X] T [ ] F : The C statement j++; is functionally equivalent to the C statement ++j; .
   When the post-increment is performed, the value of j is taken before the increment. For the pre-increment, the value of j is taken after the increment. But in both statements, the value is discarded - the only functional impact is to increment j, so functionally, both statements have the same result.

   (e) [X] T [ ] F : In a C if-then-else statement, there is an "if" keyword, and there may be an "else" keyword, but there is no "then" keyword..

   (f) [ ] T [X] F : If I code a C program in file "myProg.c", then I execute that program by typing "./myProg.c".
   The C program must be compiled first... the .c code is not executable.

   (g) [ ] T [X] F : Most of the time compiler warnings can be ignored. The compiler is just informing you that your code will work, but the compiler isn't smart enough to read your code without making some assumptions.
   Even though the compiler will produce an executable file when there are compiler warnings, chances are that there is a bug in your code that the compiler warning can help you find.

   (h) [X] T [ ] F : If the "then" block of an if statement always executes a "return" statement, an "else" block for the if statement is not needed because code after the entire if statement only gets executed if the condition is false.

   (i) [X] T [ ] F : The C compiler treats **for**(i=0;i<12;i++)sum+=i; exactly the same as **for** (i = 0 ; i < 12 ; i ++) sum + = i ; .

   (j) [X] T [ ] F : The C loop code **int** x=0; **while**(x&10); { x++; ... } is syntactically correct, but will result in an endless loop with an empty loop body.
   This is the trickiest question I've ever asked. The semicolon after the condition of the while statement indicates that there is an empty loop body, making the statement true.

2. (5 points) Check the items that are are valid literal values in C.

   [X] 3.6e1 [ ] 00800 [X] "livebeef"

   [X] '+' [ ] 0xlivebeef

3. (5 points) Check the statements that are valid declaration statements in C. (You may assume **stdbool.h** has been included.)

- [X] int SPH /* seconds per hour */ = 60*60;
- [X] char b4z='z'-1;
- [X] bool moreData=true;
- [ ] int power = // 7;
- [ ] int #validDeclare=3;

4. (5 points) For the following, put a check in front of the statements that are syntactically correct, in other words, will compile without any errors, assuming the following declare statements:

```
int n=5; char init='W'; float g=9.8; int nums;
```

- [X] for(n=0;n<6;n++) init++;
- [ ] n = n /* 3;
- [ ] if (init >= 'M') { n=0; { g=7.6; }
- [X] while(init<0) init++;
- [ ] while(for>6) g++;

5. (5 points) Check the single best answer to fill in the blank in the following statements:

(a) An assignment statement is a special kind of C expression that writes the value of the expression on the right hand side to the target specified on the left hand side.

[ ] A write [ ] A condition [ ] An if/then/else [X] An assignment [ ] A question-mark

(b) A logical binary expression like && performs a single logic evaluation on the truth values of its arguments, but a bitwise binary expression, like & performs multiple logic evaluations - one for each column of bits in its arguments.

[ ] bytewise [ ] superlogical [X] bitwise [ ] variable [ ] binary

(c) A break keyword exits from a loop in C, but a continue keyword terminates the current iteration of the loop, and goes directly to the next iteration of the loop.

[ ] leave [ ] end [X] continue [ ] restart [ ] iterate

(d) Keywords in C such as else, while, or switch disrupt the normal sequential control flow of C code.

[ ] in-order [X] sequential [ ] looping [ ] consequential [ ] alphabetic

(e) When C performs integer division using the "/" operator, if the dividend is not an exact multiple of the divisor, there will be a non-zero remainder that can be determined using the % operator.

[X] remainder [ ] percentage [ ] quotient [ ] reminder [ ] leftover

6. (5 points) Given the following declarations:

```
int a = 4; int b=-2; int c=5; float fx=3.1; float fy=-2.1;
```

Determine the value of the following expressions (Be sure to use a decimal point in floating point answers, and leave out decimal points in integer answers.)

(a) a+b*c _____-6_____

(b) (a+b)*c _____10_____

(c) b?'X':'Y' _____'X'_____

(d) (c\%b) || (c<b) _____1_____

There was a typo in this question, so everybody got it right.

(e) fx * 2.0 _____6.2_____

7. (10 points) Given the following C code:

```
int x; int n;
printf("Pick two positive intergers, x and n :> ");
scanf(" %d %d",&x,&n);
assert(n>0); // Quits if n<=0
int div=x/n;
int rem=x%n;
int y=(div*n) + rem;
```

What is the relationship between x and y? (Pick the best answer.)

☐ x < y ☒ x == y ☐ x > y ☐ Undeterminable ☐ Divide by Zero error

8. (10 points) Given the following C code:

```
int ipp=1; int ip=1;
while(ip<10) {
    int i=ipp+ip;
    printf("%d, ",i);
    ipp=ip; ip=i;
}
printf("\n");
```

Which of the following numbers will be printed? (Check **all** that apply.)

☐ 1 ☒ 2 ☐ 4 ☒ 5 ☒ 13 ☐ 21

The first iteration prints "2, ", and sets ipp=1, ip=2; the second iteration prints "3, ", ipp=2,ip=3; then "5, ", ipp=3,ip=5; "8, ", ipp=5,ip=8; finally "13, ipp=8,ip=13. Then, ip is no longer less than 10 so the loop quits. The entire print out will be "2, 3, 5, 8, 13," which represent elements of the Fibbonnacci sequence. 2 points off for every incorrect response up to 10.

Answer the following questions by filling in the blanks.

9. (10 points) Given the following C code:

```
int x=2; int i;
for(i=1;i<8; i=x%11 ) {
    x = x + i;
    if (0==x%3) continue;
    if (0==i%2) break;
}
printf("x=%d i=%d\n",x,i);
```

What get's printed to stdout?

x=19 i=4

In iteration 1, x=2+1=3; x%3=0; i=3%11=3;. In iteration 2, x=3+3=6; x%3=0, i=6%11=6. In iteration 3, x=6+6=12; x%3=0; i=12%11=1. In iteration 4, x=12+1=13; x%3=1; now x is not divisible by 3, so we do not "continue", i%2=1; and since i is not divisible by 2 we do not break yet, so i=13%11=2. In iteration 5, x=13+2=15; x%3=0; i=15%11=4;. In iteration 6, x=15+4=19; x%3 = 1; so we do not continue. i%2=0 so we break out of the loop

10. (10 points) What is wrong with the following lines of C of code? :

```
int x=0; int y=0;
for(x=0; x<7; x++) { y=y+x; }
int x=y/3;
```

The x variable may not be declared twice.

The compiler already knows that x is an int after it is declared the first time. It is illegal to specify that x is an int again.

11. (10 points) What does the following code print when compiled and executed?

```
char a = 0b01000100;
int pos=0;
char mask=0b00000001;
while(mask) {
    if (mask & a) { printf("Bit %d is on\n",pos); }
    mask <<=1; pos++;
}
```

Bit 2 is on
Bit 6 is on

12. (15 points) Write a C program to calculate the average of a list of integers, provided by your user. You will need to write a loop to ask for the next number. As long as that next number is not a zero, the loop should continue, but a zero indicates the end of the list. You may assume your user will enter at least one non-zero number. You will need to prompt the user to enter another number, and scan for that number inside the loop. Use printf("Enter the next number (0 to quit) :> "); scanf(" \%d",&num); to prompt for a new number.

Typo... should have read: scanf(" %d",&num); No points off if you copied my error.

You will need to keep track of the number of values provided by the user as well as the sum of those numbers. When the user has enterred a zero to quit, you should end the loop, and print a message that indicates the number of values in the list (not counting the final zero), and the average of those values. You may use integer division (rounding towards zero) to report the average.

Here is an example of the output from a correctly coded solution to this problem:

```
Enter the next number (0 to quit) :> 4
Enter the next number (0 to quit) :> 5
Enter the next number (0 to quit) :> 6
Enter the next number (0 to quit) :> -3
Enter the next number (0 to quit) :> 7
Enter the next number (0 to quit) :> 0
Average of 5 numbers is 3
```

```c
#include <stdio.h>
int main() {
   int num=1; int sum=0; int count=0;
   while(num!=0) { // could have been "while(1)"
      printf("Enter the next number (0 to quit) :> ");
      scanf(" %d",&num);
      if (num==0) break;
      sum+=num;
      count++;
   }
   printf("Average of %d numbers is %d\n",count,sum/count);
   return 0;
}
```

Or

```c
#include <stdio.h>
int main() {
   int num=0; int sum=0; int count=0;
    do {
      printf("Enter the next number (0 to quit) :> ");
      scanf(" %d",&num);
      sum+=num;
      if (num!=0) count++;
   } while(num!=0);
   printf("Average of %d numbers is %d\n",count,sum/count);
   return 0;
}
```

Question:	1	2	3	4	5	6	7	8	9	10	11	12	Total
Points:	10	5	5	5	5	5	10	10	10	10	10	15	100