Name: ___________________________________

1. (24 points) Given the following C function…

```
int evalGate(char gateType,int i1, int i2) {
    if (gateType=='&') {
        if (i1==0 || i2==0) return 0;
        else return 1;
    } else if (gateType=='|') {
        if (i1==0 && i2==0) return 0;
        else return 1;
    } else if (gateType=='X') {
        if (i1==i2) return 0;
        else return 1;
    }
    return -1;
}
```

   a. What value is returned for evalGate('&',1,1)? ___1___
   b. What value is returned for evalGate('|',1,1)? ___1___
   c. What value is returned for evalGate('&',1,0)? ___0___
   d. What value is returned for evalGate('Z',1,1)? ___-1___
   e. What value is returned for evalGate('X',1,1)? ___0___
   f. What value is returned for evalGate('&',1,0)? ___0___
   g. What value is returned for evalGate('X',1,7)? ___1___
   h. What value is returned for evalGate('|',12,0)? ___1___

2. (10 points) If the "else" keywords were all removed from the evalGate function above,
   a. Would the function still compile and run? _yes
   b. Would any result values change? _no_

3. (10 points) If we assume a gateType of '&' represents and AND gate, '|' represents an OR gate, and 'X' represents an exclusive-or (XOR) gate, and the C truth values of i1 and i2 are the inputs to the gate, then does the evalGate function correctly evaluate the result of the gate? If not, give an example of values for gateType, i1, and i2 which produces the wrong answer. i1=1, i2=7 evalGate(1,7)=1, XOR(T,T)=0
Could have written "if ((i1&&!i2) || (!i1&&i2)) return 1; else return 0;"

4. (20 points) Consider the following C code, which produces the output on the right:

```
int main() {
    float gravity=-9.8; // gravity is 9.8 meters per second2
    int t; // time in seconds
    float velocity[100]; float position[100];
    position[0]=500.0; velocity[0]=0.0; // Initial conditions

    for(t=1;position[t-1]>0; t++) {
        velocity[t]=velocity[t-1]+gravity;
        position[t]=position[t-1]+velocity[t];
        if (position[t]<0.0) position[t]=0.0;
    }

    float *pp=position; float *vp=velocity; t=0;
    while(*pp>0) {
        printf("At time %d height is %5.1f, velocity is %5.1f\n",
            t,*pp,*vp);
        t++; pp++; vp++;
    }
    return 0;
}
```

```
At time 0 height is 500.0, velocity is   0.0
At time 1 height is 490.2, velocity is  -9.8
At time 2 height is 470.6, velocity is -19.6
At time 3 height is 441.2, velocity is -29.4
At time 4 height is 402.0, velocity is -39.2
At time 5 height is 353.0, velocity is -49.0
At time 6 height is 294.2, velocity is -58.8
At time 7 height is 225.6, velocity is -68.6
At time 8 height is 147.2, velocity is -78.4
At time 9 height is  59.0, velocity is -88.2
```

a. What is the value of position[2]? 470.6
b. What is the value of velocity[2]? -19.6
c. What is the value of position[10]? 0.0
d. How many loops are in the code? 2
e. If pp has the value 0x0A0F C8A0 when t=0, then what will the value of pp be in hexadecimal when t=1? 0x0A0F C8A4

5. (12 points) Given the code above,

a. Would the code still work correctly if position[0] was set to 1000.0 at the beginning of the problem? If not, what would break? Yes, the code would still work

b. Would the code still work correctly if the position[0] was set to 1000000.0 at the beginning of the problem? If not, what would break? No… the velocity and position arrays would overflow and eventually cause a segmentation violation.

6. (24 points) You are an engineer who needs to design a replacement for a highway bridge. In order to find out the stress levels that the bridge must support, you have installed an electronic monitor on the existing bridge. Whenever a vehicle crosses the bridge, the monitor will provide your C function with the following information:
   - The time of day, in seconds since midnight.
   - What lane the vehicle is in: 0=northbound, 1=southbound.
   - How heavy the vehicle is: 1=compact, 2=midsize, 3=suv, 4=pickup, 5=truck, 6=semi

   a. You need to create a structure definition for the information from the monitor. Your structure definition should contain a field for the time of day, a field with an enumerated type for which lane the vehicle is in, and a field with an enumerated type for how heavy the vehicle is. Write everything you need for your structure definition here:

```
struct vehicle {
   int time;
   enum lne { northbound=0,southbound=1} lane;
   enum wgt { compact=1, midsize=2, suv=3, pickup=4, truck=5, semi=6 } weight;
};
```

   b. You need to write a function to allocate space for a new instance of your structure, as defined in part a. Write the C code for a function to get space from the heap using malloc for an instance of your structure, and return a pointer to that memory. You do not need to initialize any of the values in the structure.

```
struct vehicle * makeVehicle() {
   return (struct vehicle *) malloc(sizeof(struct vehicle));
}
```