1. Consider the following struct definitions in Reduced-C (similar to C/C++). Specify the size of each struct on a typical RISC architecture (like ieng9) or 0 if it is an illegal definition. Assume ILP-32 compiler mode.

```
structdef FOO1 {
    FOO1 * a;
    float b;
    function : void bar( FOO1 &x)
    {
        int[10] y;
    }
    int[3] c;
    int d;
};
```

```
structdef FOO2 {
    int * a;
    float b;
    function : void bar()
    {
        int x = *this.a;
    }
    FOO2 * c;
    int * d;
};
```

```
structdef FOO3 {
    int a;
    float b;
    function : void bar()
    {
        FOO3 *x;
    }
    int c;
    int[2] d;
};
```

Size _______           Size _______      Size _______

2. Using the Reduced-C Spec (which closely follows the real C language standard), given the definitions below, indicate whether each expression evaluates to either a

A) Non-Modifiable L-val     B) Modifiable L-val     C) R-val

```
function : int * foo1() { /* Function body not important. */ }
function : int & foo2() { /* Function body not important. */ }
float[9] a;
float x;
const float y = 5.5;
float *p = &x;
```

- foo2() __ 420
- *foo1() __ ++foo2() __ y __ *(int *)p
- a __ a[2] __ foo2 __ *(int *)&x __ &p __ foo1() __

3. Choosing from the following five operators,

<table>
<thead>
<tr>
<th>=    &gt;    ==   &amp;&amp;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Op1</strong> <strong>Op2</strong> <strong>Op3</strong> <strong>Op4</strong></td>
</tr>
</tbody>
</table>

and list all the operators that are valid for each of the __Op#__ below in each Reduced-C statement.

```
bool a = 2.5 __Op1__ 3;
const int b = 4 __Op2__ 2;
bool c = a __Op3__ (a __Op4__ false);
```

__Op1__ can be: ________________

__Op2__ can be: ________________

__Op3__ can be: ________________

__Op4__ can be: ________________
4. From the Reduced-C Spec (which follows closely the real C language standard), complete the following:

A) Addressable  
B) Not Addressable  
C) Modifiable  
D) Not Modifiable

A Non-Modifiable L-value is _____ and _____.

An R-value is _____ and _____.

A Modifiable L-value is _____ and _____.

5. Given the following Reduced-C code below, fill in the blanks of the compile error that should be reported according to this quarter's Project I spec. Use the letters associated with the words in the box below.

```c
typedef float F1;
typedef F1 F2;
typedef int I1;
typedef I1 I2;

I1 x;
I2 y;
F2 z;

x = z = y; // Compile error reported here. Assume this stmt is inside a function.
```

Value of type ____ not ____ to variable of type ____ .

6. State whether constant folding can be performed by the compiler according to this quarter's Reduced-C spec in the following Reduced-C statements (Y or N)

```c
function : void foo()
{
    const int a = 5;
    int b = 3;

    const int c = a + 10; ______
    int[53 + c] d; ______
    d[-2 + (a * b)] = c; ______
    b = d[d[2] + c]; ______
    int e = d[a + c]; ______
    e = d[13 + b]; ______
    e = d[e + a]; ______
    d[5 - 2 + c] = e; ______
}
```

7. In Reduced-C (which follows closely the real C standard) we use ______________ name equivalence with struct types. And we use ______________ name equivalence with typedefs. For all other types we use ______________ equivalence.