1. For the following Reduced-C code, indicate whether each statement will result in a compile-time error or not:

(A) No error  (B) Compile-Time Error  [11 pts]

```c
struct def FOO { int x; int y[2]; }
struct def BAR { int x; int y[2]; }

FOO a;

function : int & foo(float & x) {
    FOO b;
    BAR c;
    b = a;
    c = a;
    b.y[1] = 4;
    b.y[2] = 8;
    b.x = b.y[1]++;
    x = b.x;
    return b.x++;
}

function : void main() {
    FOO d;
    float e;
    a.x = foo(e) = 9;
    a.x = foo(foo(e));
    d.x = a.x;
    d.y = a.y;
}
```

2. Identify the following C constructs as either:  (A) Pure Declaration  (B) Definition  [6 pts]

```c
struct baz * foo( int x );
int * foo( int x );
extern int ******** p;
extern int a[50][50];
extern int fooz( int x );
struct fubar { int x; } s1;
```

3. Give an example of a non-converting type cast (underlying bit pattern does not change):  [3 pts]

```c
float f = 5.00;
int  i = ____________________________;
```
4. Given the following pseudocode, determine the program output based on the specified scoping rule: [20 pts]

```c
int x = 0;        -- global var declaration
int y = 0;        -- global var declaration

void sum_xy( int a, int b )
    x = x + a;
    y = y + b;

void beta()
    int y = 2;    -- local var declaration
    sum_xy( 6, 6 );
    cout << x << endl;
    cout << y << endl;

void alpha()
    int x = 8;    -- local var declaration
    sum_xy( 7, 7 );
    cout << x << endl;
    cout << y << endl;
    sum_xy( 3, 3 );
    beta();
    cout << x << endl;
    cout << y << endl;

sum_xy( 1, 1 );
alpha();
beta();
cout << x << endl;
cout << y << endl;
```

What does the program output if the language uses **static** scoping?

What does the program output if the language uses **dynamic** scoping?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>