1. Show the memory layout of the following C struct definition taking into consideration the SPARC data type memory alignment restrictions discussed in class. Fill bytes in memory with the appropriate struct member/field name. For example, if member/field name `p` takes 4 bytes, you will have 4 `p`'s in the appropriate memory locations. If the member/field is an array, use the name followed by the index number. For example, some number of `p[0]`s, `p[1]`s, `p[2]`s, etc. Place an X in any bytes of padding. Structs and unions are padded so the total size is evenly divisible by the most strict alignment requirement of its members. [10 pts]

```c
struct foo {
    int a[3];
    double b;
    short c[1];
    double *d;
    char e;
    short f;
    char g;
};

struct foo fubar;
```

What is the `offsetof(struct foo, f)`? ________

What is the `sizeof(struct foo)`? ________

If `struct foo` had been defined as `union foo` instead, what would be the `sizeof(union foo)`? ________

2. Given the C array declaration

```c
short a[12];
```

Mark with an A the memory location(s) where we would find `a[10]` [4 pts]

Mark with an A the memory location(s) where we would find `a[10]`

(over)
3. Consider the following struct definitions in Reduced-C (similar to C/C++). Specify the size (in bytes) of each struct on a typical RISC architecture (like ieng9) or 0 if it is an illegal definition. [6 pts]

```c
structdef LARRY {
    int    a;
    float* b;
    float[4] c;
    LARRY[4] d;
    function : void see()
    {
        int* e;
    }
};

structdef CURLY {
    int    a;
    float[4] b;
    float* c;
    CURLY* d;
    function : void hear()
    {
        int[50] e;
    }
};

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

Size _______  Size _______  Size _______

4. Using Reduced-C syntax, define an array of array of array of floats named baz such that

```c
baz[3][17][6];
```

is the last element in the data structure. This will take three lines of Reduced-C code [5 pts]

5. Using the Rt-Lt Rule, give the C variable declaration for a variable named halloween that is a pointer to a function that takes a char* and a double as two parameters and returns a pointer to a pointer to an array of 16 elements where each element is a pointer to a short. [5 pts]