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]`, `p[1]`, `p[2]`, 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. [14 pts]

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

struct doge shibe;
```

What is the `offsetof( struct doge, g )`? ________

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

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

2. Given the C array declaration

```c
short a[2][7];
```

Mark with an A the memory location(s) where we would find `a[1][4]` [6 pts]

(a box represents a byte in memory)

If `a[0][0]` is allocated at memory location 6000 decimal, what value does `&a[1][3]` evaluate to? ________ (over)
3. Using Reduced-C syntax, complete the below statements in order to define variable `yuno` such that it refers to a memory layout corresponding to the following ragged array of integers:

This will be done without dynamic memory allocation and using 3 temporary variables, resulting in a total of seven statements of Reduced-C code [14 pts]

```c
_______ tmp0 _______; // define tmp0
_______ tmp1 _______; // define tmp1
_______ tmp2 _______; // define tmp2
_______ yuno _______; // define yuno
__________ = __________; // connect yuno to tmp0
__________ = __________; // connect yuno to tmp1
__________ = __________; // connect yuno to tmp2
```

4. Using the Right-Left Rule, give the C variable declaration for a variable named `valentine` that is a pointer to a pointer to an array of 14 elements where each element is a pointer to a function that takes a `short` as a parameter and returns a pointer to a `struct february`. [6pts]