Homework 6

Struct and union

Please answer the following questions according to the definition of the union.

```
union ele {
    struct s1 {
        char cc;
        union ele *next;
        short ss;
        long long int li;
    } e1;
    int i;
    struct s2 {
        char c;
        struct s1 (*f) (int i, short ss, long long int li);
        char str[3];
        short s;
        int *p[2];
        char c2;
        int ii;
    } e2;
} u;
```

1. Fill in the following blocks. (please represent address with Hex)

sizeof(u.e1)	32
sizeof(u.e2)	48
sizeof(union ele)	48
u	0x601060
u.e1.next	0x601068
u.e1.li	0x601078
u.e2.f	0x601068
u.e2.p[1]	0x601080

2. How many bytes are WASTED in struct s2 under x86-64? If you can rearrange the declarations in the struct s2, how many bytes of memory can you SAVE in struct s2 compared to the original declaration under x86-64?

48 – (1+8+3+2+16+1+4) = 13 bytes wasted.
8 bytes. It will use 40 bytes. (5 byte padding at the end of struct)

Pointers and array

Answer following questions and explain why. Assume we use x86-64 machines.

1. Is the value of &(a[1]) equals to value of (b+1)?
   ```
   int a[2]; char *b = a;
   ```
   No, sizeof(int) is 4, sizeof(char) is 1.

2. Is the value of &(a[1]) equals to value of (b+1)?
   ```
   int a[2]; char **b = a;
   ```
   No, sizeof(int) is 4, sizeof(char) * is 8.

3. Is the value of &(a[1]) equals to value of (b+1)?
   ```
   int *a[2]; char **b = a;
   ```
   Yes, both a and b are pointer to pointers.

4. Is the value of &(a[1]) equals to value of (b+1)?
   ```
   int a[2]; char (*b)[2][2] = a;
   ```
   Yes, b is a pointer to a 2D array, and the size of this 2D array is 4 bytes.

5. Is the value of &(a[1]) equals to value of (b+1)?
   ```
   int a[2]; char (**b)[2][2] = a;
   ```
   No, b is a pointer points to a pointer to a 2D array, so b+1 is 8 byte-advanced than b.

6. What is a?
   ```
   int *(*a[3])(int *, int);
   ```

An Array with 3 elements points to a function with two parameters (int * and int) returning int pointer.