Homework 2 - CECS 424
Taina Coleman - 012535859

1. For the output Statically (S) and Dynamically (D) of the code presented in the question, I have:

	S	D
first()	1	1
printx()	1	1
second()	2	2
printx()	2	1

And the difference is because static scoping binding is defined by the physical structure of the program and dynamic scoping binding happens on runtime.

2.
   - Language design time
   - Compile time
   - Link time
   - Link time
   - Runtime

3. a) 24 bytes, because d, e and f are going to be destroyed when the subroutine is over.
   b) Static links, where each frame points to the "parent" frame.

4. The output for Shallow binding (S) and Deep binding (D) is:

	S	D
1	1, 0	1, 0
2	2, 0	5, 2
3	3, 0	0, 0
4	4, 0	4, 4

The difference is that Deep Binding binds the environment at the time the procedure is passed as an argument, and Shallow Binding binds the environment at the time the procedure is actually called.

5.

- First, late binding gives the code a greater chance to be version-independent. Second, it might compile faster, since it has less references thus a smaller file size than early binding.

- 1.
```
int a, *pa;
*pa = 4;
free(pa);
a = *pa; //pa exists, but it doesn't point to a valid memory cell
```

2.
```
int func (int *p){
    int value = *p++;
    free(p);
    return value;
}
int main(){
    int *p = malloc(sizeof(int));
    func(*p);
    printf("%p", *p);
}//dangling pointer because it has been freed in func
```

3.
```
char* func(char a){
    char b;
    char *c;
    b = a;
    c = &b;
    return c;
}//the returning pointer will be a dangling reference since c doesn't exist out of the subroutine
```