1 Sample Problems

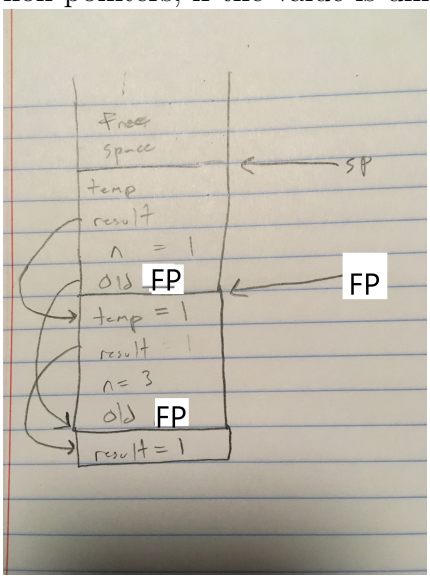
Here are some sample problems to give you an idea of what I might ask on the midterm:

1. (pts) Consider the following C program:

```
void fib(int n, int *result) {
    int temp;
    if (n < 2) {
        *result = n;
    } else {
        fib(n-1, result);
        fib(n-2, &temp);
        *result += temp;
    }
}
```

```
void main() {
    int result;
    fib(3, &result);
    printf("%d\n", result);
}
```

Draw the memory stack for the moment when main has called fib(3, ...), which has called fib(1, ...), which is just about to return. Fill in the values on the stack: for pointers, draw the pointer, for non-pointers, write the values. For both pointers and non-pointers, if the value is unknown leave a blank entry.



2. (pts) Consider the following code in a new language we are creating:

```
int f(x, y) {
    x = y + x
    return x
}

main() {
    int x = 1
    int y = 1
    int z = f(x,y) // This is line A
    print(x + y + z)
}
```

If we decide our language will pass parameters using "call-by-value", does the call to f(x, y) in line A exhibit referential transparency? What if we decide to use "call-by-reference"?

If we do call by value, then the line does exhibit referential transparency. The function only receives copies of the values, and so the function has no side effects. So replacing the expression with its value does not change the program.

If it is call by reference, then the line does not exhibit referential transparency. The assignment that occurs inside the function will change the value of x in the main function, so if we replace the expression with its value the final value of x will be different.

3. Consider the following code, written in a language that supports dynamic dispatch:

```
class Student {
    public void Print() {
        System.out.println("Student");
    }
}

class CSE extends Student {
    public void Print() {
        System.out.println("CSE");
    }
}

class Main{
    public static void PersonPrint( Student o ) {
        o.Print();
    }
    public static void main(String[] args) {
        Student x = new Student();
        Student y = new CSE();
        CSE z = new CSE();
        PersonPrint(x);
        PersonPrint(y);
        PersonPrint(z);
    }
}
```

Identify the static and dynamic targets for the invocation o.Print() in PersonPrint. What is printed when we run the program?

The static target of o.Print() is the Print method from Student.

The first time o.Print() is executed, the dynamic target will be the Print method in Student.

The second time o.Print() is executed, the dynamic target will be the Print method in CSE.

The third time o.Print() is executed, the dynamic target will be the Print method in CSE.

This program prints:

Student

CSE

CSE

4. (pts) Suppose in my C program I have the following code:

```
int *ptr, *qtr;
int a = 1, b = 2, count;
ptr = &a;
qtr = &b;
count = *ptr + *qtr;
```

What can you say about the value of count?
Pointer ptr gets the memory address of a, and pointer qtr gets the memory address of b. So then count = 1 + 2 = 3.