Bryan Duong

013104973

CECS 424 Assignment 12

1. (5 points) Elaborate on the reasons why even though parameters passed in registers may sometimes need to have locations in the stack.

   - The stack is meant to be used to store function calls and the associated parameters related to that function. So, when a function is called, the stack allocates space for the function and the parameters it needs to use. This results in how parameters having locations in the stack despite already being stored in a register.

2. (10 points) Consider the following function and call:

```
procedure f(x, y, z)
   x := x + 1
   y := z
   z := z + 1
. . .
i := 1; a[1] := 10; a[2] := 11
f(i, a[i], i)
print(i, a[1], a[2])
```

   Determine the outputs of the program if arguments are passed 1) by value, 2) by reference, and 3) by name.

1) By value
   i = 1; a[1] = 10; a[2] = 11
   1, 10, 11

2) By reference
   i = 3; a[1] = 2; a[2] = 11
   3, 2, 11

3) By name
   i = 3; a[1] = 10; a[2] = 11
   3, 10, 2

3. (10 points) Check out the output of the following program in `C` in your system and suggest an explanation in terms of activation records in the stack.

```
void foo() {
   int i;
   printf("%d ", i++);
}

int main() {
   int j;
   for (j = 1; j <= 10; j++) {
      foo();
   }
}
```

- Given the scope integer i was declared in, integer i is considered a dynamic variable. Because of this instead of being declared to 0, some other indeterminate behavior occurs from the operating system. Integer i was never initialized but was still allocated space to the stack. Because of this, integer i was initialized by the operating system. So, the values that appear on the terminal are the values that are the operating system gives to integer i.