Andre Maldonado
CS 354-1
TA2-1
29 October 2019

3.2)

Super-duper.scm would not work if allocation was on the stack. The heap is the ideal storage location for lists, character strings and sets. The accessing order would be thrown off

```
(define (e-duper element cnt)
        (if (>cnt 1)
                (cons(car element) (e-duper element ( - cnt 1)))
                        element
        )
)

(define (super-duper source count)
        (if (list? source)
                (if(pair? source)
                        (e-duper (cons (super-duper ( car source ) count)
(super-duper (cdr source) count )) count)
                source
        )
        source
        )
)
```

3.4)

a. (C)

```
main(void){
        printf("%s", "example code");
}

void foo(void){
        int x = 10;
}
```

The variable x is static, live but not in scope unless the execution is not inside the function

b. (C++)

Functions both sub and add use the same named variables in order to do their calculations. The variable name 'r' is live in both functions but the only one in scope is sub because it is called in the main.

```
int main(){
        int s = sub(4,3);
        cout<<s;
}
Int add( int a, int b){
        int r;
        r= a+b;
        return r;
}

Int sub( int c, int d){
        int r;
        r= a-b;
        return r;
}
```

c. (java)

The variable b is live but never in scope called unless foo is called. It has no use until it's called.

```
public static void main(String args[]){
        Int f = foo(x);
        System.out.print(f);
}

public int foo(int a){
        int b = 10;
        return (a* b)+1;
}
```

3.5)

The program prints 1,1 then 1,1 then 1,2. C# would likely not throw any errors because like Java, it uses static scope. The called variables would not conflict with each other because they only hold their values in the block they where they were instantiated.

3.7)

a. The issue is that Java conveniently allocates and frees memory automatically via the garbage-collector. C does not have this automation so the use of malloc is required to avoid memory leaks. Brad has a memory leak because his list will fill and pass the memory limit, thus crashing.

b. When delete_list is called, the references point to old data which was corrupted by having a reverse list.

3.14)

With static scope, the program prints 1,1,2,2. With dynamic scope it prints 1,1,2,1. It depends on whether the program sees the x assignment from first() or second().

3.18)

With shallow binding the program would print 10 20 30 40 because set_x and print_x would always access the local x in foo. With deep binding the program prints 10 52 00 44 because set_x accesses the global x when n is even and local x when n is odd. Print_x accesses the global x when n is 3 or 4 and the local foo x when n is 1 or 2.