Learning Outcome: Students will gain experience in writing assembly programs that respect x86 standard calling conventions.

Write an assembly program for each question.

- You will need to follow the X86 standard calling convention in this lab. This means that your assembly program should be similar to the ones generated by C compilers or debuggers.
- Use the **.data** section to declare string literals containing format specifiers such as "%s" and "%c\n".
- Local variables (indicated **red** in each provided code) must be allocated in the **stack frame** of the function where they are defined. Do not use the data section for local variables.

1. (30 points)

```
int main(){
    char x[20];
    scanf("%s", x);
    for(char* curr = x; *curr != 0; curr++){
        printf("%c\n", *curr);
    }
}
```

2. (40 points)

```
void swap(int* x, int* y) {
       int temp = *x;
       *x = *y;
       *y = temp;
}

int main() {
       int a, b;
       a = 10;
       b = 20;
       swap(&a, &b);
       printf("a = %d, b = %d\n", a, b);
       return 0;

}
```

3. (30 points)

```
int* array_copy(int* array, int length) {
	int* copy = (int*)malloc(length * 4);
	for (int i = 0; i < length; i++) {
		copy[i] = array[i];
	}
	return copy;
}
int main() {
	int array[3] = {10, 20, 30};
	int* copy = array_copy(array, 3);
	array[0] = 40;
	printf("%d, %d, %d\n", copy[0], copy[1], copy[2]);
}
```

Deliverables:

a. You must submit working code to your Assignment12 directory before 11/17/2021 at 7 PM.

NAME:

1. (30 points) The student could use the stack memory for a local variable array, store characters onto the stack, and iterate over the characters.

2. (40 points) The student wrote an assembly program that `swaps` two integers in a separate function following the x86 standard calling conventions.

3. (30 points) The student could declare an integer array using the stack memory and return a copy of an array to `main` following the x86 standard calling conventions.