CISC 235 Assignment 1

Output of Each Function

Function 1:

```
Using Recursive Function:

n = 7:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 7
n = 18:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 7 14 28 9 18
n = 19:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 44 88 29 58 19
n = 22:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22
n = 43:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 7 14 28 56 112 37 74 148 49 98 196 65 130 43
```

```
Using Non-Recursive Function

n = 7:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 7
n = 18:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 7 14 28 9 18
n = 19:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 44 88 29 58 19
n = 22:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22
n = 43:
1 2 4 8 16 5 10 20 40 13 26 52 17 34 11 22 7 14 28 56 112 37 74 148 49 98 196 65 130 43
```

Function 2:

```
Using Recursive Function:

n = 7:
2 4 7
n = 18:
2 4 6 4 2 5 8 12 18
n = 19:
2 4 6 4 2 5 8 12 19
n = 22:
2 4 7 4 3 2 4 6 9 14 22
n = 43:
4 3 2 4 6 9 14 3 2 4 6 9 2 4 6 4 2 5 8 12 18 28 43
```

```
Using Non-Recursive Function

n = 7:
2 4 7
n = 18:
2 4 6 4 2 5 8 12 18
n = 19:
2 4 6 4 2 5 8 12 19
n = 22:
2 4 7 4 3 2 4 6 9 14 22
n = 43:
4 3 2 4 6 9 14 3 2 4 6 9 2 4 6 4 2 5 8 12 18 28 43
```

Function 3:

```
Using Recursive Function:

n = (0, 7):
0 1 2 3 4 5 6 7
n = (1, 18):
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
n = (4, 19):
4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
n = (-1, 22):
-1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
Using Non-Recursive Function

n = (0, 7):
0 1 2 3 4 5 6 7
n = (1, 18):
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
n = (4, 19):
4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
n = (-1, 22):
-1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

Function 4:

```
Using Recursive Function:

n = (0, 7):
0 2 1 4 7 6 5 3
n = (1, 18):
1 3 2 5 8 7 6 4 10 13 12 11 15 18 17 16 14 9
n = (4, 19):
4 6 5 8 10 9 7 12 14 13 16 19 18 17 15 11
n = (-1, 22):
0 -1 3 2 1 6 5 9 8 7 4 12 11 15 14 13 18 17 20 22 21 19 16 10
```

```
Using Non-Recursive Function

n = (0, 7):
0 2 1 4 7 6 5 3
n = (1, 18):
1 3 2 5 8 7 6 4 10 13 12 11 15 18 17 16 14 9
n = (4, 19):
4 6 5 8 10 9 7 12 14 13 16 19 18 17 15 11
n = (-1, 22):
0 -1 3 2 1 6 5 9 8 7 4 12 11 15 14 13 18 17 20 22 21 19 16 10
```

Part 3

In part one I chose to double the size of the array every time it gets filled up. This was not the best choice considering there were never many elements stored in the stack at the same time. Therefore, the number of elements contained in the stack at the same time rarely surpassed 10 and when it did it wouldn't surpass it by much. Since I allocated 10 extra spaces whenever it filled up there were many spaces left untouched. I should have just added a small amount of space (e.g. 1 or 2 additional spaces) every time the array filled up.