Design:
My Enumerative top-down synthesizer is designed as the below pseudo code:

```
synthesize (cfg, examples) {
    workListQueue.add(startSymbolAST)
    while (worklist is not empty) {
        AST = workListQueue.remove()
        if (AST is complete)
            if (AST program is correct on examples) {
                return AST program
            }
        }
        else {
            workListQueue.addAll(Expand(AST,cfg))
        }
    }
    return null
}
```

- Uses a queue to implement the worklist and therefore it performs a BFS style search
- Expand function returns all expanded trees of AST containing all possible expansions of the first non-terminal symbol of AST

Tests:
AST Depth 1 programs:

Examples	Program	Result

x=2, y=2, z=3 -> 4
x=4, y=3, z=7 -> 7
x=6, y=5, z=5 -> 11
x=8, y=1, z=5 -> 9
	Add(y, x)	Pass

x=2, y=2, z=3 -> 4
x=4, y=3, z=7 -> 12
x=6, y=5, z=5 -> 30
x=8, y=1, z=5 -> 8
	Multiply(y, x)	Pass

AST Depth 2 programs:

Examples	Program	Result

x=2, y=2, z=3 -> 4
x=3, y=3, z=5 -> 6
x=6, y=5, z=7 -> 7
x=3, y=6, z=5 -> 5
	Ite(Lt(3, y), z, Add(x, x))	Pass

x=2, y=2, z=3 -> 20
x=4, y=4, z=7 -> 88
x=6, y=6, z=5 -> 132
x=8, y=1, z=5 -> 54
	Multiply(Add(z, y), Add(y, x))	Pass

AST Depth 3 programs:

Examples	Program	Result

x=2, y=2, z=3 -> 18
x=3, y=4, z=5 -> 80
x=6, y=5, z=7 -> 245
x=3, y=6, z=5 -> 120
	Multiply(z, Add(y, Multiply(y, x)))	Pass

x=4, y=2, z=6 -> 23
x=2, y=4, z=5 -> 31
x=6, y=5, z=7 -> 68
x=1, y=8, z=5 -> 51
	Add(3, Multiply(y, Add(z, x)))	Pass

Issues:

- Without allocating extra heap space, the program seems to run out of memory when trying to synthesize some depth 3 programs and most depth 4+ programs

Additional Comments:

- When running the synthesizer please use the -Xmx8g option in the execution command to increase heap size