VM1 - take VM code and output Assembly

Test this out in the CPU Emulator (don't need the VM Emulator)

Two folders (with subfolders) in Project 7:

```
        MemoryAccess
                BasicTest
                PointerTest
                StaticTest
        StackArithmetic
                SimpleAdd
                StackTest
```

These folders hold .vm files and .tst files

Notes:

```
        Delete comments for testing
        Want to make sure that the last thing in your Assembly file is the infinite loop
(END)
@END
0;JMP
```

My recommendation is to do things in this order:

```
SimpleAdd
BasicTest
PointerTest
StaticTest
StackTest
```

But you could do

```
SimpleAdd
StackTest
BasicTest
StaticTest
StackTest
```

Make sure you get SimpleAdd working and verified with a test before continuing on.

```
char* push(char* memorySegment, char* value)
```

```
push constant 7
```

output might be:

```
"@7\nD=A\n@SP\nA=M\nM=D\n@SP\nM=M+1\n"
```

Another way would be to send in a pointer to the outputFile and write directly to that file

Inside of that function, you need if statements to decide what the memorySegment is

```
if(strcmp(memorySegment,"constant") == 0))
```

Create a similar function for the add() code

Add doesn't take any parameters (unless you're sending in the outputFile)

```
char* add()
```

That would get you through SimpleAdd

After that, to work on BasicTest, you need to create a sub() function

Make sure you're subtracting in the right order

```
push 41
```

```
push 1
sub

you should get 40 in 256, not -39

Memory areas:
local - RAM 1 LCL
argument - RAM 2 ARG
this - RAM 3 THIS
that - RAM 4 THAT
pointer
temp
static

Remember that local 1 means 1 plus the value in local

let's say local is 300
local 0 means RAM[300]
local 1 means RAM[301]

push means put something from a memorySegment onto the stack

push local 0

would put whatever is in RAM[300] on the stack

This was my Python helper function for this:
def translateSegment(name):
    if name == "local":
        return "LCL"
    elif name == "argument":
        return "ARG"
    elif name == "this":
        return "THIS"
    elif name == "that":
        return "THAT"

For local, I would manually set SP to 256, set RAM 1 to 300 and put some value in RAM[300]

Then see if you can get that value in RAM[256]

---
pointer means 3 + value

push pointer 5

means push the value at RAM[8]

push pointer 21

means push the value at RAM[24]

YOU DO NOT LOOK in RAM[3], you just add 3 to whatever the value is
---

temp is the same as pointer, except you add 5 instead of adding 3

---
static is the hard one
when you see static, it wants you to translate it to
@nameOfFile.value

so if you are reading
example.vm
and see
```

static 5

that should become
@example.5

push should be normal, just have the translation

for pop, that adds an extra step
for my pop, I had to do
@example.5
D=A

Just to be clear, the final submission must be in C
but Python is okay for prototyping

It's OK if you want to do the Python prototyping and then send me your .py file to help debug