Question 6: We’re deep, deep undercover… (10 pts, 25 min)

You've been contracted by top secret government agencies to make a really quick, super-portable message encoder in MIPS! Go spy go! You have to implement an algorithm for encoding an ASCII zero-terminated (C-Style) string. Your algorithm should be based on the following C code:

```
void encryptThis(char* cleartext, int* cypher, int* cyphertext_buffer){
  if(*cleartext == ‘\0’){
      *cyphertext_buffer = 0;
  }else{
      *cyphertext_buffer = *cypher + *cleartext;
      encryptThis(++cleartext, ++cypher, ++cyphertext_buffer);
  }
}
```

Implement the above C function **in a non-recursive manner** in MIPS. Don't clobber any registers that you shouldn't, **`$a0`** corresponds to **`cleartext`**, **`$a1`** corresponds to **`cypher`**, and **`$a2`** corresponds to **`cyphertext_buffer`**. You can assume that you will only be passed the usual types of ASCII values (in the range 0-127). Use as few lines as possible (you may not need to use every blank).

encryptThis:

Name: ___________________________ Login: cs61c-____

Question 7: Meet my friend Andy Anderson… (10 pts, 25 min)

```
Main: ......
      # Set up $a0
      jal foo
      ......
foo:  li $v0, -1
      lbu $t0, 0($a0)
      bne $0, $t0, done
      addi $sp, $sp, -8
      sw $ra, 4($sp)
      sw $t0, 0($sp)
      addi $a0, $a0, 1
      jal foo
      lw $t0, 0($sp)
      and $v0, $v0, $t0
      lw $ra, 4($sp)
      addi $sp, $sp, 8
done: jr $ra
```

```
                foo(                          )
{




   return                  ;
}
```

a) What does the function `foo` return?

b) In the box above, fill in the C code for the function `foo`. Be sure to include arguments and return values, along with their types.

c) If we call your function `foo` like this: **`printf("%c", foo("Cal"));`** What will be printed?

d) What would `foo` do if we changed its first line to read "**`li $v0, 0`**"?