Reverse Polish Notation (RPN) Calculator

This project involves creating a RPN Calculator that is able to read Programs from a File. This involves Stacks (Linked Lists) and a Hash Table (Separate Chaining).

This calculator will read in a one-line program from a file. This line read from the file will then be passed into a pre-written tokenizer, which will take every individual piece of the program and turn them into tokens. In every step of the calculator, the calculator will get the next token and you will process it by either pushing it into a stack, or by executing the operation the token represents. This calculator will work with both number values and variables, which will be stored as Symbols with a variable name and a value in a symbol table (hash table). The calculator will process the following types of tokens that may be in a program.

- Value: A number in the expression
- Variable: A name that is in an expression
- Operator: A mathematical operator. We will have: +, -, /, and *
- Assignment: This token to assign a value to a variable. This is an = in the equation
- Print: This token tells the calculator to print out the current stack top.

The Code written contains functions from three files:

- **rpn.c** The main calculator logic code to run the programs and perform the file I/O.
- **stack.c** The stack implementation. Singly linked-list with a dummy node.
- **hash.c** The symbol table implementation. Separate Chaining with Rehashing.

1. rpn.c:

static int read_file(char *filename, char *line)

- This function opens the file for reading, line into line (up to MAX_LINE_LEN) and then closes the file.

static int parse_token(Symtab *symtab, Stack_head *stack, Token *tok)

- Parses each token in the file just read
- Function receives a pre-initialized symtab and stacks and the next Token tok to parse
- This function is used to help generate the output to match this documentation

2. stack.c:

Stack_head *stack_initialize()

- Function mallocs a new Stack_head and initializes it. Sets its count to 0 and next pointer to NULL and returns the new Stack_head pointer

void stack_destroy(Stack_head *head)

- This function frees an entire Stack (nodes and head).

int stack_push(Stack_head *stack, Token *tok)

- This function pushes a Token on to the stack.
- Creates a new Node using node_create and passing in tok to it
- Insert this new Node at the top of the stack adjusts the counter.

Token *stack_pop(Stack_head *stack)

- This function pops and returns a Token from the stack
- Removes the top Node from the stack and returns a pointer to the Token
- Returns the Token pointer

Token *stack_peek(Stack_head *stack)

- This function returns the top Token pointer on the stack and returns the Token pointer

int stack_is_empty(Stack_head *stack)

- This function returns 1 if the stack is empty or 0 otherwise

3. hash.c:

Symtab *hash_initialize()

- This function mallocs a new Symtab and initializes it
- Allocates space for a new table in the symtab on the heap dynamically.
- Array type is a double pointer and each index holds a Symbol Pointer and is an array of linked lists, and initializes all elements to NULL, sets capacity
- Returns the new Symtab Pointer

void hash_destroy(Symtab *symtab)

- Function destroys an entire Symtab by going through each index of the table and frees linked list symbols in there

int hash_get_capacity(Symtab *symtab)

- This function returns the current number of indexes total in the table

int hash_get_size(Symtab *symtab)

- This function returns the current number of Symbols in the table

int hash_put(Symtab *symtab, char *var, int val)

- This will either add a new variable to the hash table or update an existing one
- Uses hash_code and modulus to get the index when hashing
- If variable already exists in the hash table, updates its val to the new val
- Checks to see if the load is >= 2.0 i.e load is size / capacity.

- If load >= 2.0, doubles the capacity and rehashes using the rehash function
- From here creates the new Symbol using symbol_create and inserts it at the index

Symbol *hash_get(Symtab *symtab, char *var)

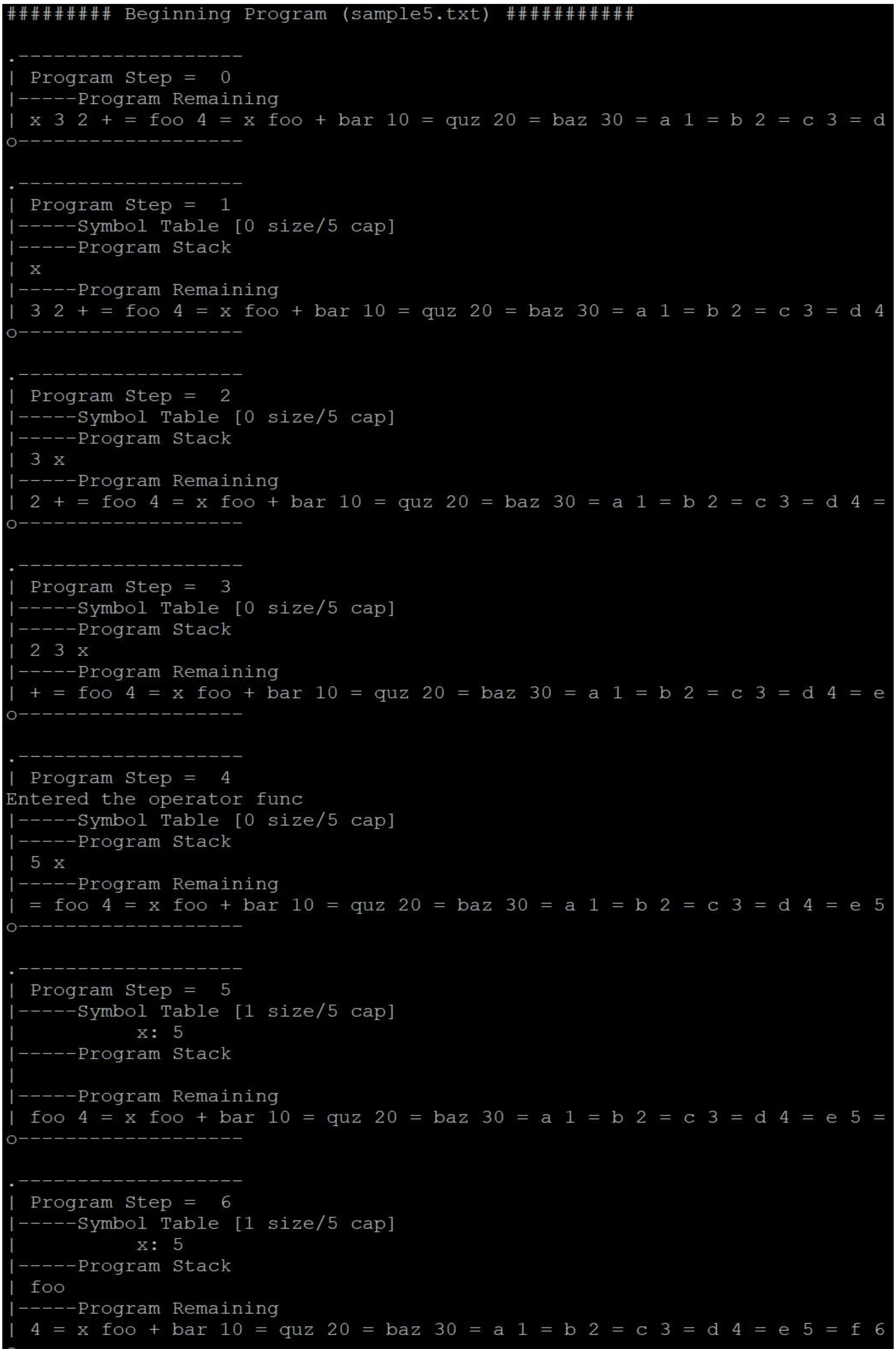
- This will return a copy of the Symbol found in the Hash Table (or NULL if not found)
- Uses hash_code and modulus to get the index when hashing
- Checks the linked list at the index for a Symbol with a matching var and if it is found, uses the symbol_copy function to copy and returns a pointer to the copy

void hash_rehash(Symtab *symtab, int new_capacity)

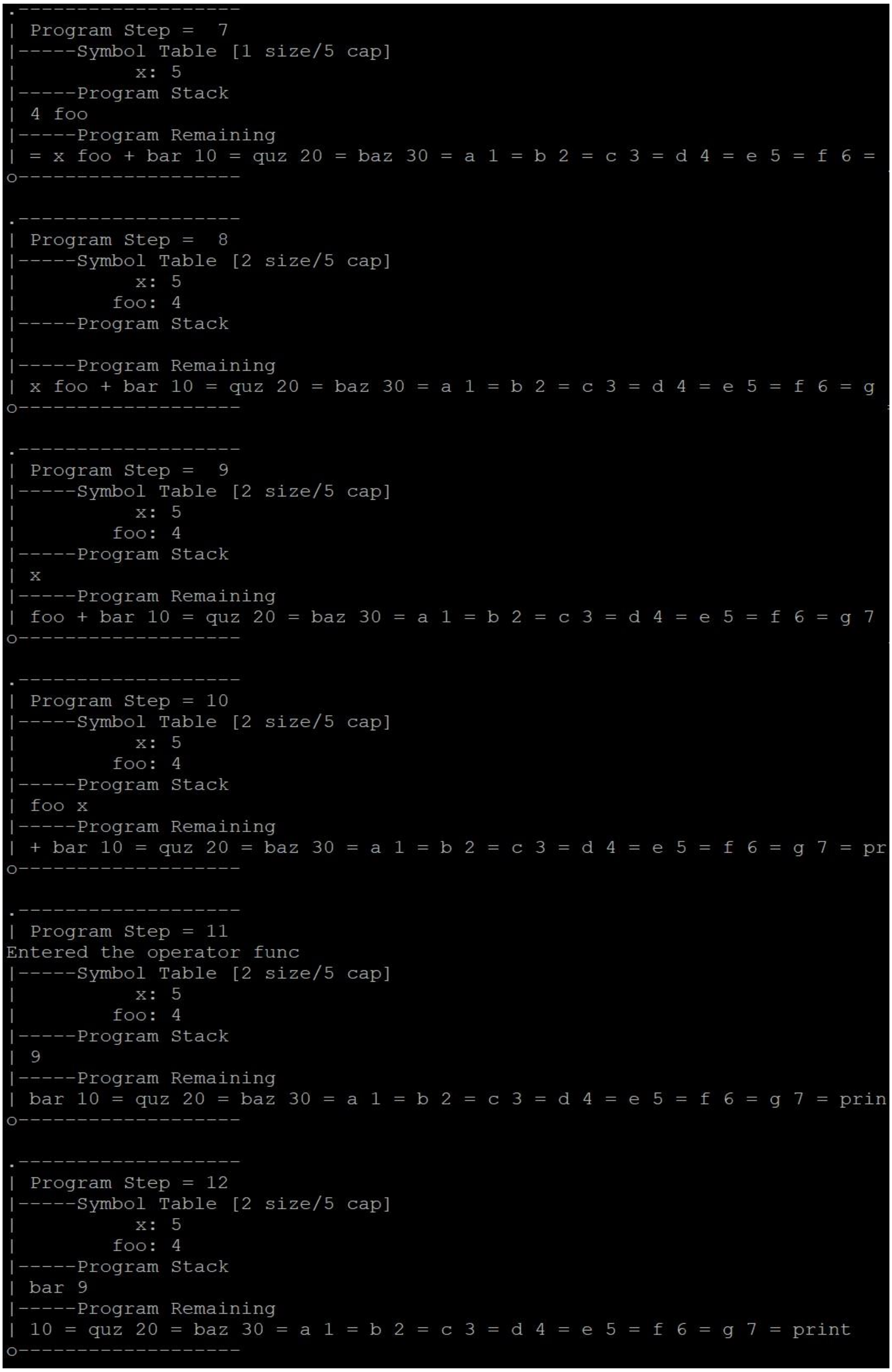
- This will double the capacity in the symtab's table so calls this with 2* the old capacity
- It is a new table and re-hashes each Symbol into it and walks through the old table and put each Symbol's data into the new table
- Frees all of the old Symbols and the old Table and update the symtab pointer to new symtab

OUTPUT

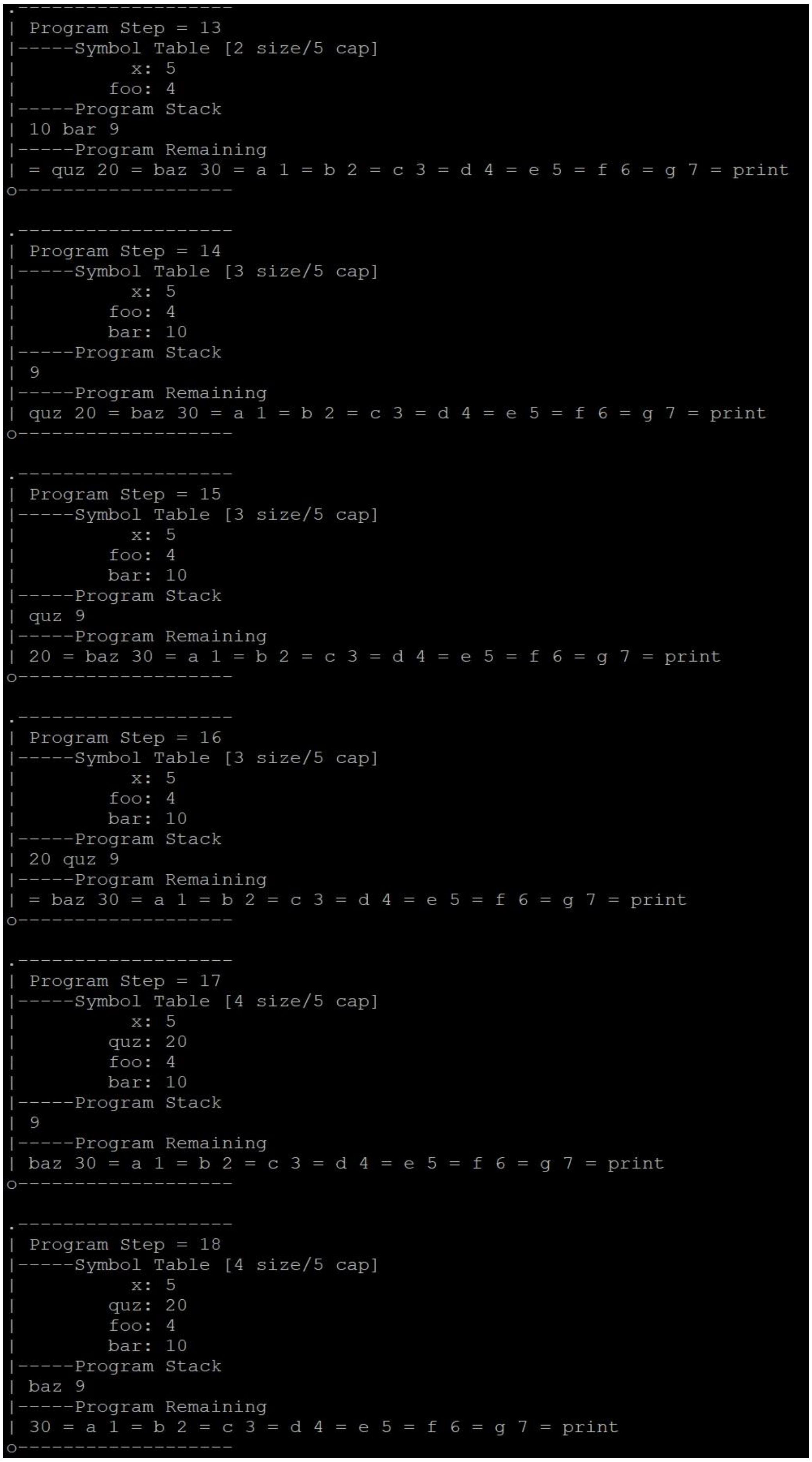
The following run is a combination of variables, values, assignments and finally a print of the output.



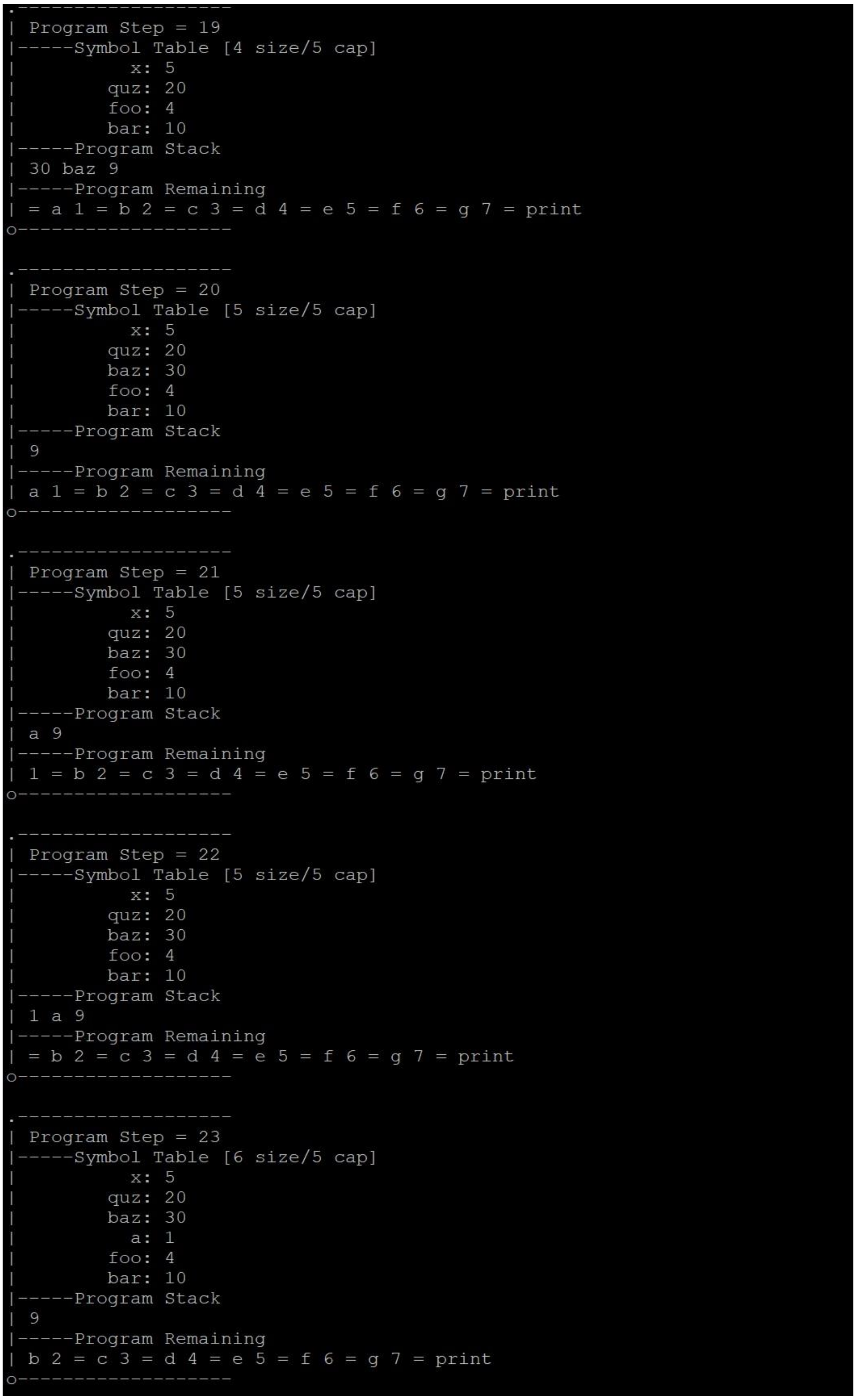
```
######### Beginning Program (sample5.txt) ###########

.-------------------
| Program Step =  0
|-----Program Remaining
| x 3 2 + = foo 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d
o-------------------

.-------------------
| Program Step =  1
|-----Symbol Table [0 size/5 cap]
|-----Program Stack
| x
|-----Program Remaining
| 3 2 + = foo 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4
o-------------------

.-------------------
| Program Step =  2
|-----Symbol Table [0 size/5 cap]
|-----Program Stack
| 3 x
|-----Program Remaining
| 2 + = foo 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 =
o-------------------

.-------------------
| Program Step =  3
|-----Symbol Table [0 size/5 cap]
|-----Program Stack
| 2 3 x
|-----Program Remaining
| + = foo 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e
o-------------------

.-------------------
| Program Step =  4
Entered the operator func
|-----Symbol Table [0 size/5 cap]
|-----Program Stack
| 5 x
|-----Program Remaining
| = foo 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5
o-------------------

.-------------------
| Program Step =  5
|-----Symbol Table [1 size/5 cap]
|         x: 5
|-----Program Stack
|
|-----Program Remaining
| foo 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 =
o-------------------

.-------------------
| Program Step =  6
|-----Symbol Table [1 size/5 cap]
|         x: 5
|-----Program Stack
| foo
|-----Program Remaining
| 4 = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6
```



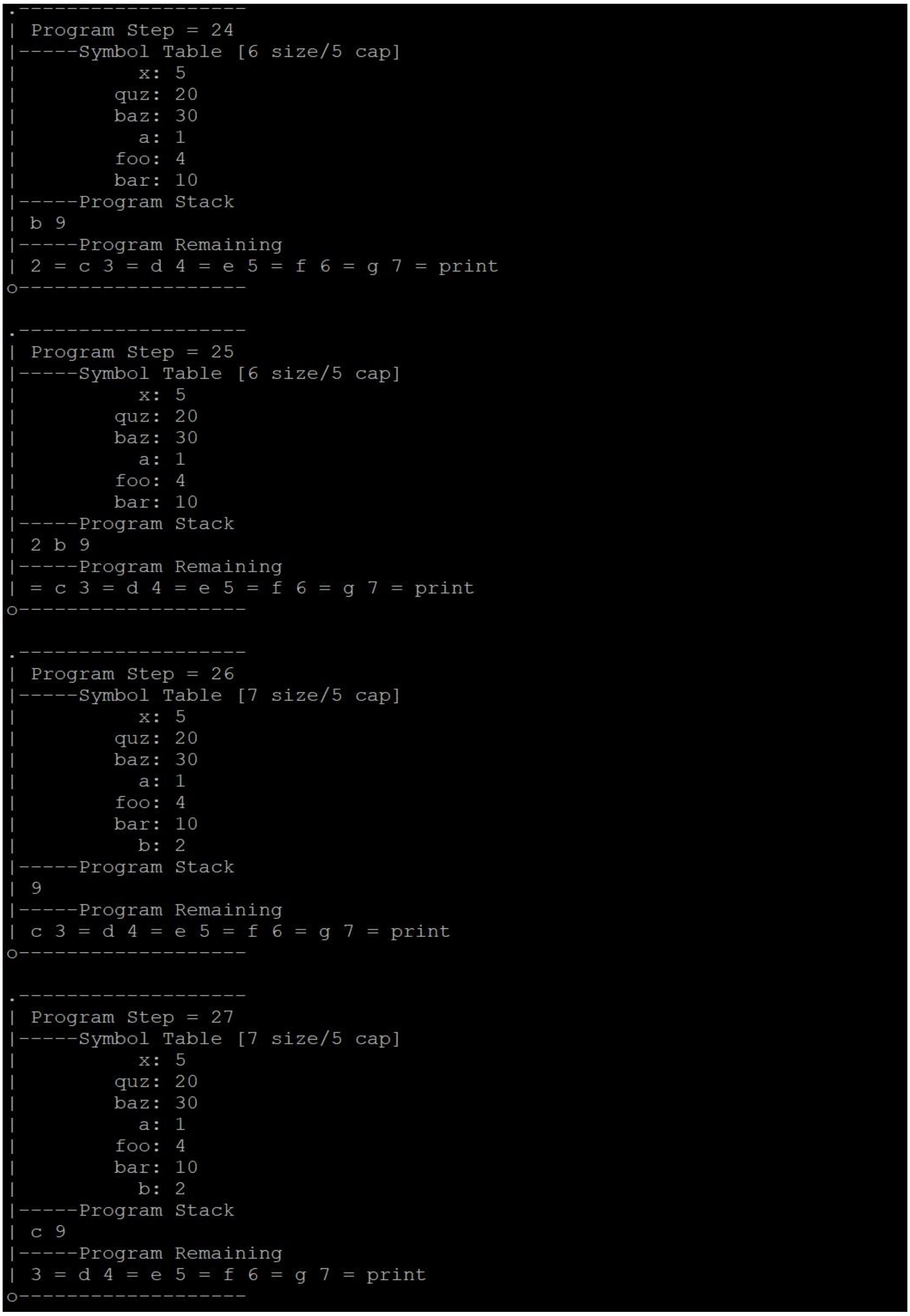
```
.-------------------
| Program Step =  7
|-----Symbol Table [1 size/5 cap]
|          x: 5
|-----Program Stack
| 4 foo
|-----Program Remaining
| = x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 =
o-------------------

.-------------------
| Program Step =  8
|-----Symbol Table [2 size/5 cap]
|          x: 5
|        foo: 4
|-----Program Stack
|
|-----Program Remaining
| x foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g
o-------------------

.-------------------
| Program Step =  9
|-----Symbol Table [2 size/5 cap]
|          x: 5
|        foo: 4
|-----Program Stack
| x
|-----Program Remaining
| foo + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7
o-------------------

.-------------------
| Program Step = 10
|-----Symbol Table [2 size/5 cap]
|          x: 5
|        foo: 4
|-----Program Stack
| foo x
|-----Program Remaining
| + bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = pr
o-------------------

.-------------------
| Program Step = 11
Entered the operator func
|-----Symbol Table [2 size/5 cap]
|          x: 5
|        foo: 4
|-----Program Stack
| 9
|-----Program Remaining
| bar 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = prin
o-------------------

.-------------------
| Program Step = 12
|-----Symbol Table [2 size/5 cap]
|          x: 5
|        foo: 4
|-----Program Stack
| bar 9
|-----Program Remaining
| 10 = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------
```



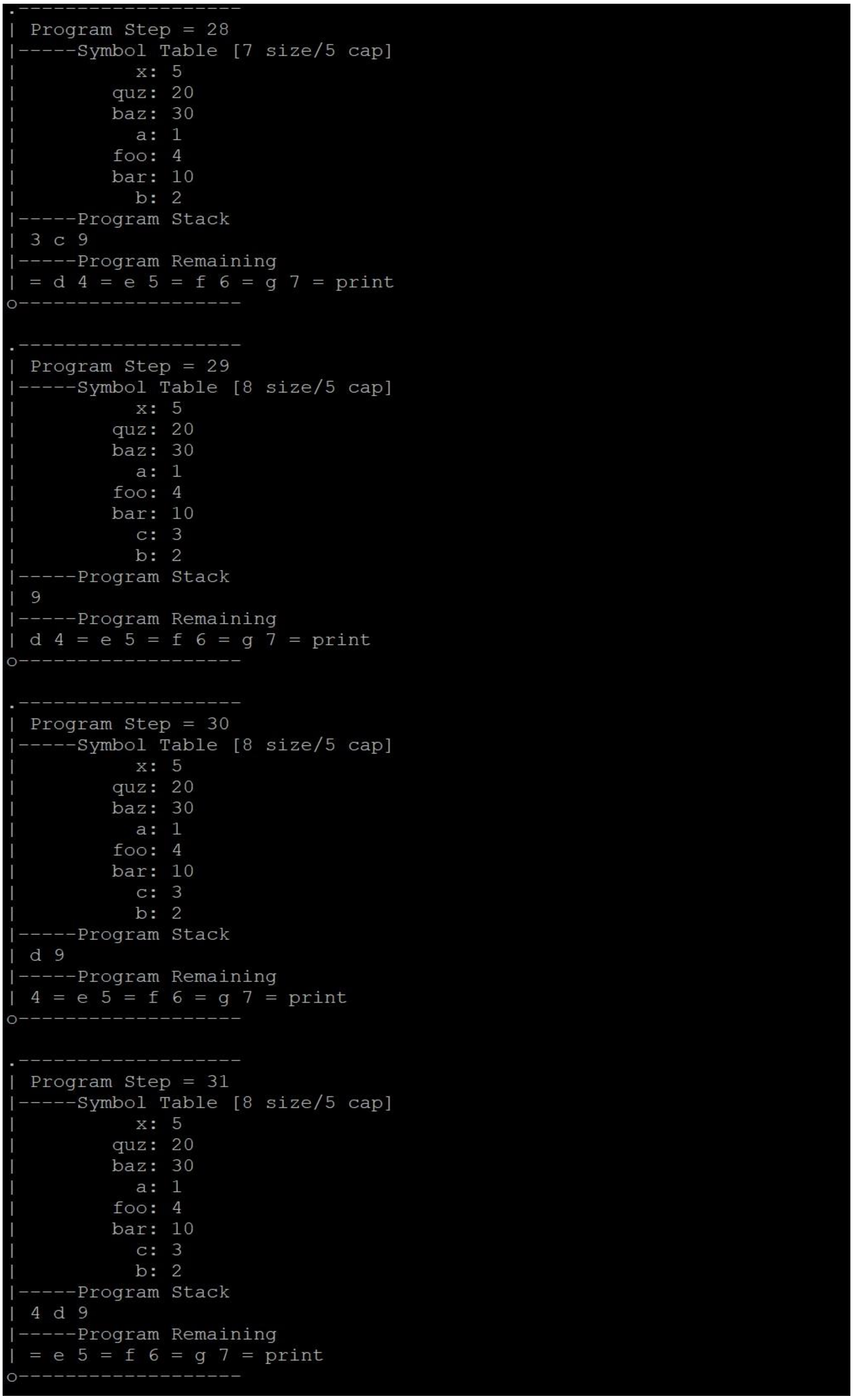
```
.-------------------
| Program Step = 13
|-----Symbol Table [2 size/5 cap]
|          x: 5
|        foo: 4
|-----Program Stack
| 10 bar 9
|-----Program Remaining
| = quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 14
|-----Symbol Table [3 size/5 cap]
|          x: 5
|        foo: 4
|        bar: 10
|-----Program Stack
| 9
|-----Program Remaining
| quz 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 15
|-----Symbol Table [3 size/5 cap]
|          x: 5
|        foo: 4
|        bar: 10
|-----Program Stack
| quz 9
|-----Program Remaining
| 20 = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 16
|-----Symbol Table [3 size/5 cap]
|          x: 5
|        foo: 4
|        bar: 10
|-----Program Stack
| 20 quz 9
|-----Program Remaining
| = baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 17
|-----Symbol Table [4 size/5 cap]
|          x: 5
|        quz: 20
|        foo: 4
|        bar: 10
|-----Program Stack
| 9
|-----Program Remaining
| baz 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 18
|-----Symbol Table [4 size/5 cap]
|          x: 5
|        quz: 20
|        foo: 4
|        bar: 10
|-----Program Stack
| baz 9
|-----Program Remaining
| 30 = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------
```



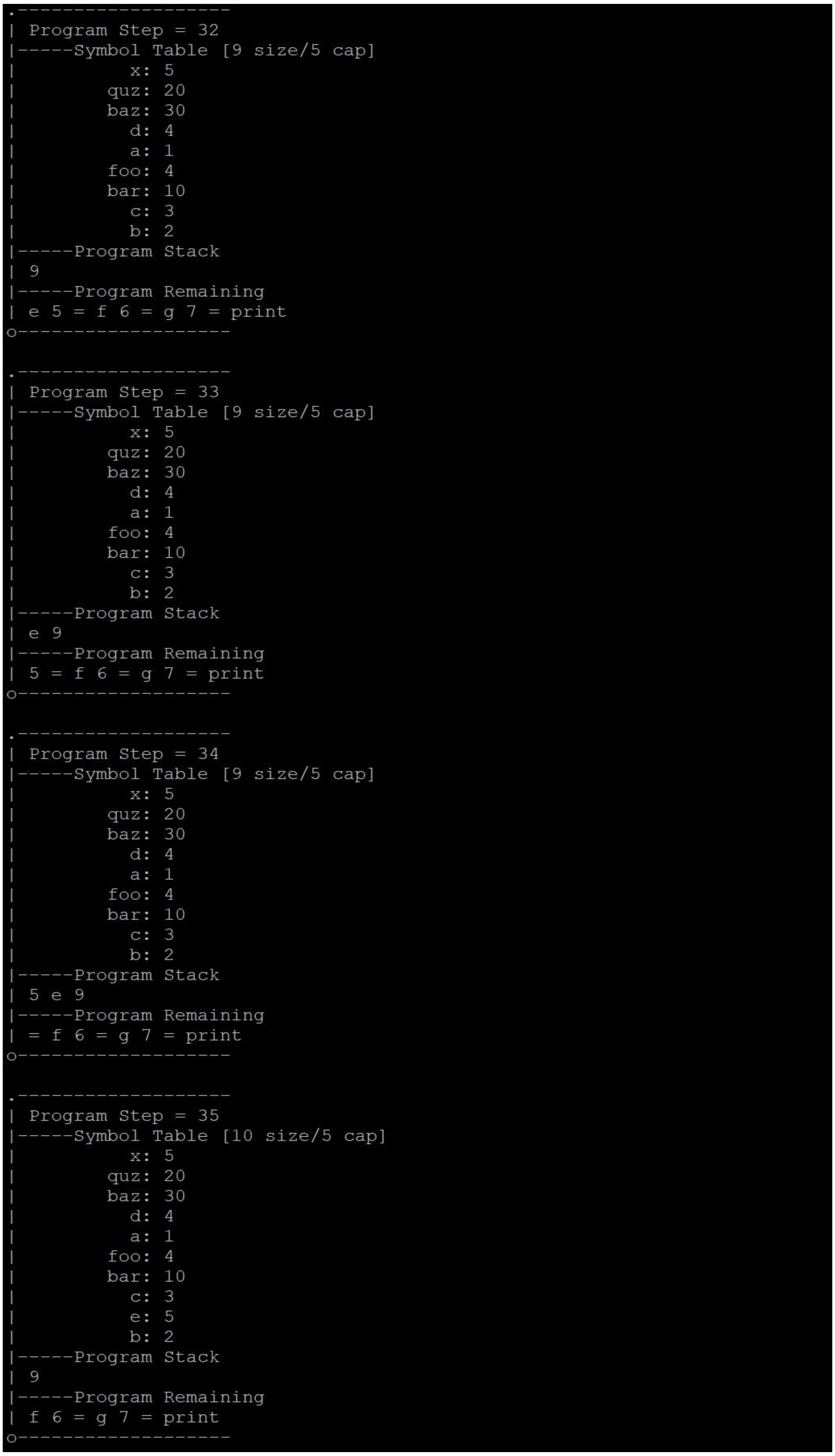
```
.-------------------
| Program Step = 19
|-----Symbol Table [4 size/5 cap]
|          x: 5
|        quz: 20
|        foo: 4
|        bar: 10
|-----Program Stack
| 30 baz 9
|-----Program Remaining
| = a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 20
|-----Symbol Table [5 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|        foo: 4
|        bar: 10
|-----Program Stack
| 9
|-----Program Remaining
| a 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 21
|-----Symbol Table [5 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|        foo: 4
|        bar: 10
|-----Program Stack
| a 9
|-----Program Remaining
| 1 = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 22
|-----Symbol Table [5 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|        foo: 4
|        bar: 10
|-----Program Stack
| 1 a 9
|-----Program Remaining
| = b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 23
|-----Symbol Table [6 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          a: 1
|        foo: 4
|        bar: 10
|-----Program Stack
| 9
|-----Program Remaining
| b 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------
```



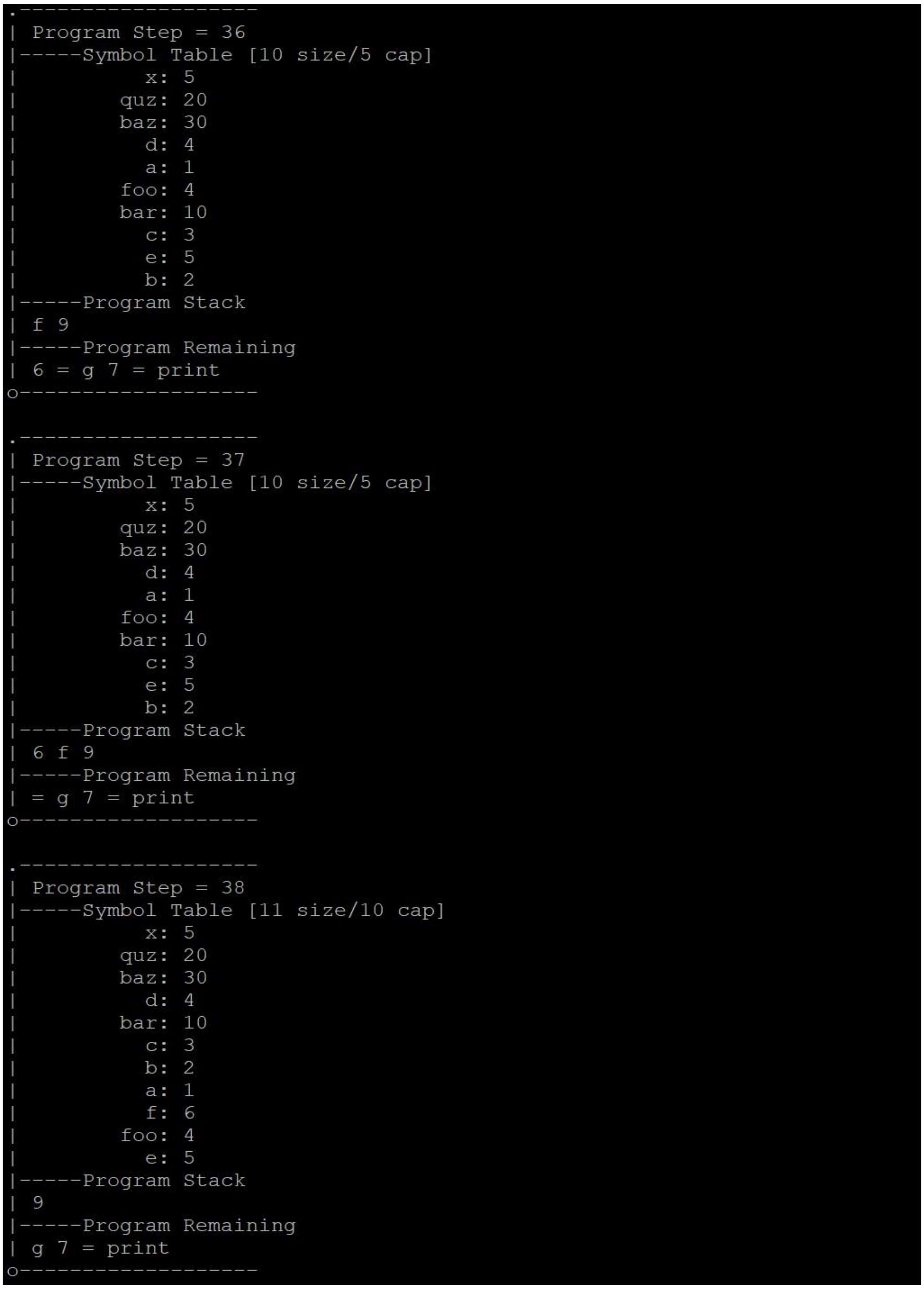
```
.-------------------
| Program Step = 24
|-----Symbol Table [6 size/5 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           a: 1
|         foo: 4
|         bar: 10
|-----Program Stack
| b 9
|-----Program Remaining
| 2 = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 25
|-----Symbol Table [6 size/5 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           a: 1
|         foo: 4
|         bar: 10
|-----Program Stack
| 2 b 9
|-----Program Remaining
| = c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 26
|-----Symbol Table [7 size/5 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           a: 1
|         foo: 4
|         bar: 10
|           b: 2
|-----Program Stack
| 9
|-----Program Remaining
| c 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 27
|-----Symbol Table [7 size/5 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           a: 1
|         foo: 4
|         bar: 10
|           b: 2
|-----Program Stack
| c 9
|-----Program Remaining
| 3 = d 4 = e 5 = f 6 = g 7 = print
o-------------------
```



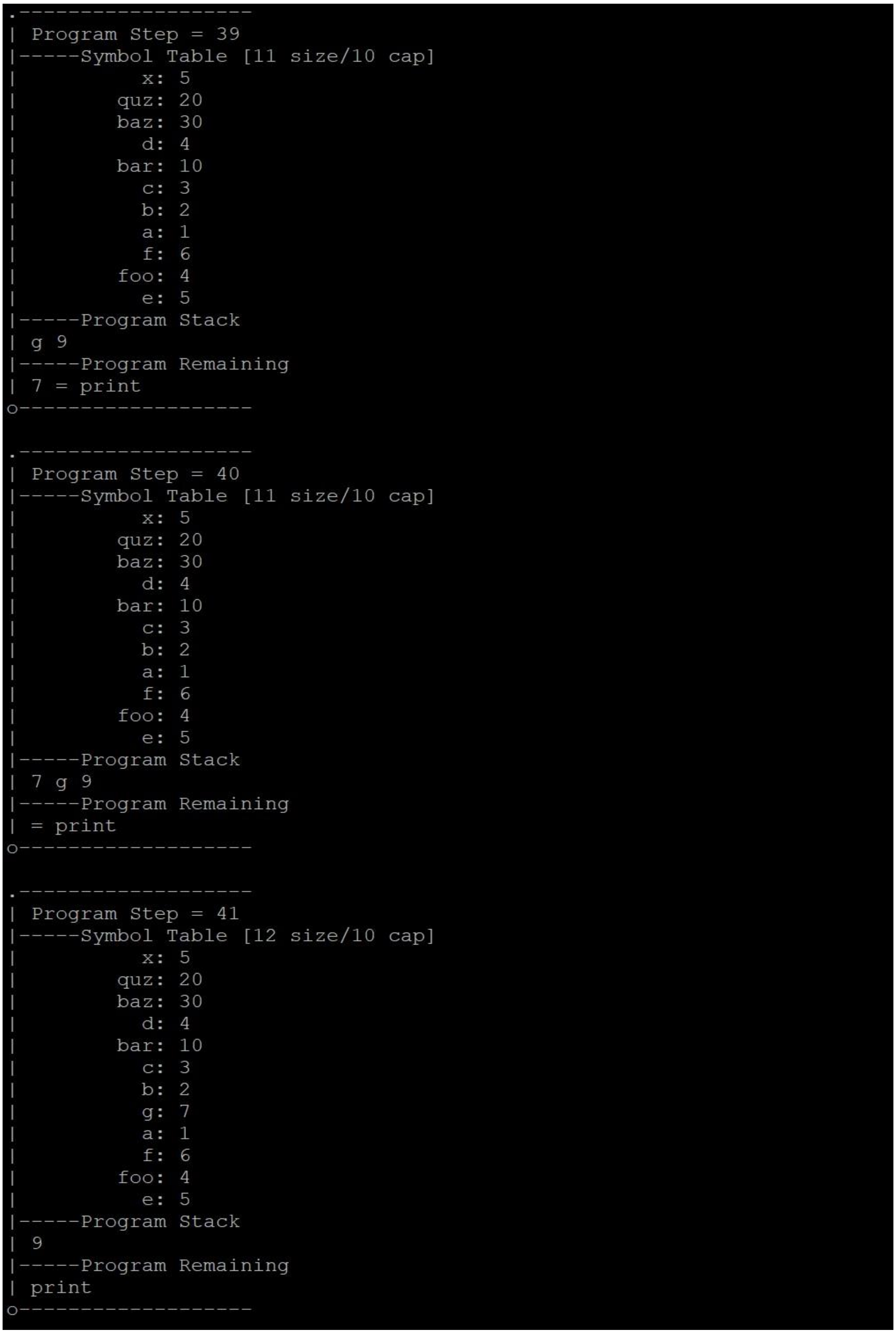
```
.-------------------
| Program Step = 28
|-----Symbol Table [7 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          a: 1
|        foo: 4
|        bar: 10
|          b: 2
|-----Program Stack
| 3 c 9
|-----Program Remaining
| = d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 29
|-----Symbol Table [8 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          b: 2
|-----Program Stack
| 9
|-----Program Remaining
| d 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 30
|-----Symbol Table [8 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          b: 2
|-----Program Stack
| d 9
|-----Program Remaining
| 4 = e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 31
|-----Symbol Table [8 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          b: 2
|-----Program Stack
| 4 d 9
|-----Program Remaining
| = e 5 = f 6 = g 7 = print
o-------------------
```

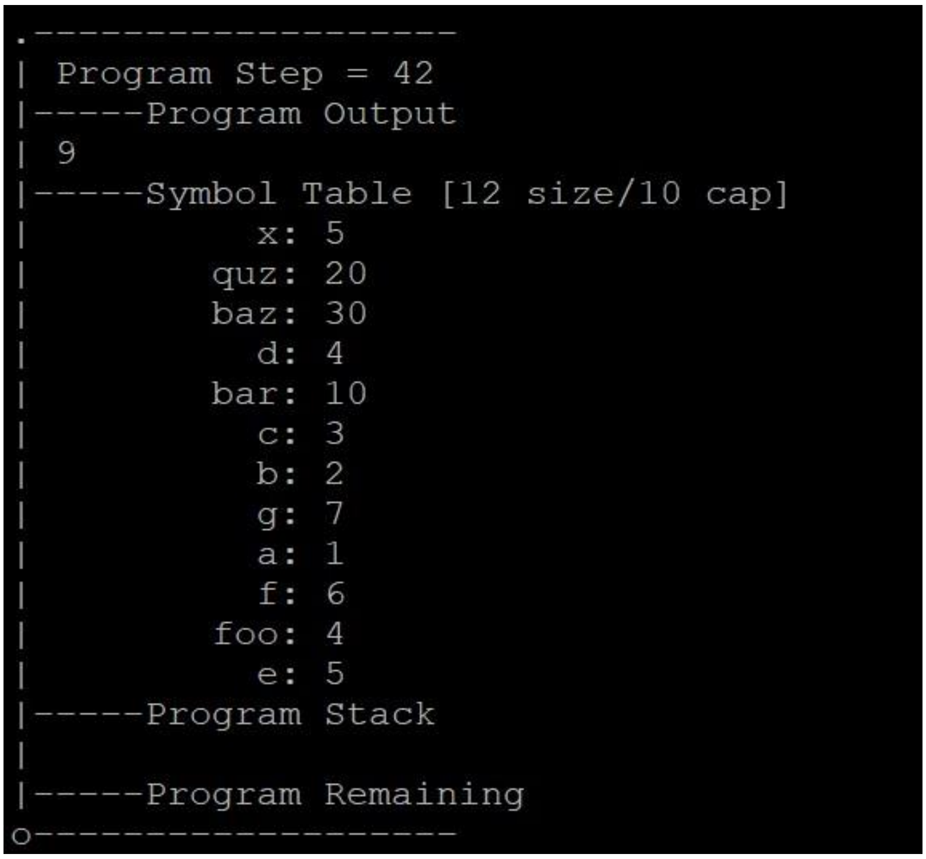


```
.-------------------
| Program Step = 32
|-----Symbol Table [9 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          b: 2
|-----Program Stack
| 9
|-----Program Remaining
| e 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 33
|-----Symbol Table [9 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          b: 2
|-----Program Stack
| e 9
|-----Program Remaining
| 5 = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 34
|-----Symbol Table [9 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          b: 2
|-----Program Stack
| 5 e 9
|-----Program Remaining
| = f 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 35
|-----Symbol Table [10 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          e: 5
|          b: 2
|-----Program Stack
| 9
|-----Program Remaining
| f 6 = g 7 = print
o-------------------
```



```
.-------------------
| Program Step = 36
|-----Symbol Table [10 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          e: 5
|          b: 2
|-----Program Stack
| f 9
|-----Program Remaining
| 6 = g 7 = print
o-------------------

.-------------------
| Program Step = 37
|-----Symbol Table [10 size/5 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|          a: 1
|        foo: 4
|        bar: 10
|          c: 3
|          e: 5
|          b: 2
|-----Program Stack
| 6 f 9
|-----Program Remaining
| = g 7 = print
o-------------------

.-------------------
| Program Step = 38
|-----Symbol Table [11 size/10 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|        bar: 10
|          c: 3
|          b: 2
|          a: 1
|          f: 6
|        foo: 4
|          e: 5
|-----Program Stack
| 9
|-----Program Remaining
| g 7 = print
o-------------------
```


```
.-------------------
| Program Step = 39
|-----Symbol Table [11 size/10 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           d: 4
|         bar: 10
|           c: 3
|           b: 2
|           a: 1
|           f: 6
|         foo: 4
|           e: 5
|-----Program Stack
| g 9
|-----Program Remaining
| 7 = print
o-------------------

.-------------------
| Program Step = 40
|-----Symbol Table [11 size/10 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           d: 4
|         bar: 10
|           c: 3
|           b: 2
|           a: 1
|           f: 6
|         foo: 4
|           e: 5
|-----Program Stack
| 7 g 9
|-----Program Remaining
| = print
o-------------------

.-------------------
| Program Step = 41
|-----Symbol Table [12 size/10 cap]
|           x: 5
|         quz: 20
|         baz: 30
|           d: 4
|         bar: 10
|           c: 3
|           b: 2
|           g: 7
|           a: 1
|           f: 6
|         foo: 4
|           e: 5
|-----Program Stack
| 9
|-----Program Remaining
| print
o-------------------
```



```
.-------------------
| Program Step = 42
|-----Program Output
| 9
|-----Symbol Table [12 size/10 cap]
|          x: 5
|        quz: 20
|        baz: 30
|          d: 4
|        bar: 10
|          c: 3
|          b: 2
|          g: 7
|          a: 1
|          f: 6
|        foo: 4
|          e: 5
|-----Program Stack
|
|-----Program Remaining
o-------------------
```