Source File: `~/4301/12/lab12.asm`
Input: Under control of `main` function
Output: Under control of `main` function
Value: 2

For $\Sigma = \{a, b, c\}$, construct an assembly function that accepts a string of unknown length as its argument and determines if the string is a member of the set consisting of

$$\left\{ \begin{array}{l} x | x \in \{a, b, c\}^+ \text{ and} \\ \quad \text{the number of } a\text{'s plus the number of } b\text{'s plus twice the number of } c\text{'s is divisible by six.} \end{array} \right\}$$

Valid strings include `aaaaaa`, `bbbbbb`, `ccc`, `babac`, `caabb`, `bcbaa`, `abcba`, `ccaa`, `cabc`, `cccccc`, `aacbbccc`, `ccccccccc`, `aaaaaaaaaaaabbbbbbbbbbbb`, and `aaabbb` but not ϵ, `a`, `b`, `aa`, `ab`, `ba`, `bb`, `aaa`, `bbb`, `abba`, `abcb`, or `abbbbaa`.

A sample `main` function for testing your function is shown in Figure 1, a template for starting the assembly function is shown in Figure 2, and a sample execution sequence is shown in Figure 3.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int isValid(char *s);
5
6  int main()
7  {
8    char s[81];
9
10   while (fgets(s, sizeof(s), stdin))
11   {
12     // remove trailing newline from s
13     s[strcspn(s, "\n")] = 0;
14     printf("%s is %s\n", s, (isValid(s) ? "valid" : "invalid"));
15   }
16
17   return 0;
18 }
```

Figure 1. `/usr/local/4301/src/lab12main.c`

```
;;; Your Name
;;; CS 3304
;;; Lab 12

;-----------------------------------------------------------------------
global  isValid:function
; Returns true if its argument is valid and false otherwise.
; HLL prototype: int isValid(char *s);
; Implements the following HLL function:
;      int isValid(char *s)
;      {
;        int sum = 0;
;        for ( ; *s != '\0'; s++)
;        {
;          if (*s == 'a' || *s == 'b')
;            sum += 1;
;          else if (*s == 'c')
;            sum += 2;
;        }
;        return sum % 6 == 0;
;      }
; Receives: one 32-bit pointer containing the address of a string
;           via the system stack
; Returns:  a 32-bit integer in EAX
;-----------------------------------------------------------------------
isValid:
SECTION .text
        push    ebp
        mov     ebp,esp
        sub     esp,4           ; create space on stack for one 32-bit int
        pushad                  ; push eax, ecx, edx, ebx, esp, ebp, esi, edi
; Stack looks like
;
;       | param *s    |   [ebp + 8]
;       | return addr |   [ebp + 4]
;       | ebp         |   ebp
;       | local var   |   [ebp -  4]
;       | eax         |   [ebp -  8] from pushad
;       | ecx         |   [ebp - 12] from pushad
;       | edx         |   [ebp - 16] from pushad
;       | ebx         |   [ebp - 20] from pushad
;       | esp         |   [ebp - 24] from pushad
;       | ebp         |   [ebp - 28] from pushad
;       | esi         |   [ebp - 32] from pushad
;       | edi         |   [ebp - 36] from pushad
;
        mov     dword [ebp-4],0 ; clear accumulator
        mov     esi,[ebp+8]     ; place addr of param in esi

```

Figure 2. `/usr/local/4301/src/lab12tmpl.asm` (Part 1 of 2)

```
50 ;
51 ; Put your code here
52 ;
53
54         popad                   ; pop edi, esi, ebp, esp, ebx, edx, ecx, eax
55         mov     eax,[ebp-4]     ; move local accumulator to eax
56         add     esp,4           ; clean local var from stack
57         pop     ebp
58         ret
```

Figure 2. `/usr/local/4301/src/lab12tmpl.asm` (Part 2 of 2)

```
1  newuser@csunix ~> cd 4301
2  newuser@csunix ~/4301> mkdir 12
3  newuser@csunix ~/4301> cd 12
4  newuser@csunix ~/4301/12> # Copy the sample input and output files
5  newuser@csunix ~/4301/12> cp /usr/local/4301/data/12/* .
6  newuser@csunix ~/4301/12> ls
7  00.dat  00.out
8  newuser@csunix ~/4301/12> # Copy the main function
9  newuser@csunix ~/4301/12> cp /usr/local/4301/src/lab12main.c .
10 newuser@csunix ~/4301/12> ls
11 00.dat  00.out  lab12main.c
12 newuser@csunix ~/4301/12> # Copy the template for the assembly code
13 newuser@csunix ~/4301/12> # and save it as lab12.asm
14 newuser@csunix ~/4301/12> cp /usr/local/4301/src/lab12tmpl.asm lab12.asm
15 newuser@csunix ~/4301/12> # Edit lab12.asm to complete the code
16 newuser@csunix ~/4301/12> # Compile the C main function
17 newuser@csunix ~/4301/12> gcc -m32 -c lab12main.c
18 newuser@csunix ~/4301/12> # Assemble the assembly function
19 newuser@csunix ~/4301/12> nasm -f elf32 -o lab12.o lab12.asm
20 newuser@csunix ~/4301/12> # Invoke the linker to create the executable
21 newuser@csunix ~/4301/12> gcc -m32 -o lab12 lab12main.o lab12.o
22 newuser@csunix ~/4301/12> ls
23 00.dat  00.out  lab12  lab12.asm  lab12.o  lab12main.c  lab12main.o
24 newuser@csunix ~/4301/12> # Execute lab12 on dataset 00
25 newuser@csunix ~/4301/12> ./lab12 < 00.dat
26 aaaaaa is valid
27 bbbbbb is valid
28 ccc is valid
29 babac is valid
30 caabb is valid
31 bcbaa is valid
32 abcba is valid
33 ccaa is valid
34 cabc is valid
35 cccccc is valid
36 aacbbccc is valid
```

Figure 3. Commands to Compile, Assemble, Link, & Run Lab 12 (Part 1 of 2)

```
37 ccccccccc is valid
38 aaaaaaaaaaaabbbbbbbbbbb is valid
39 aaabbb is valid
40  is invalid
41 a is invalid
42 b is invalid
43 aa is invalid
44 ab is invalid
45 ba is invalid
46 bb is invalid
47 aaa is invalid
48 bbb is invalid
49 abba is invalid
50 abcb is invalid
51 abbbbaa is invalid
52 newuser@csunix ~/4301/12> ./lab12 < 00.dat > my.out
53 newuser@csunix ~/4301/12> diff 00.out my.out
54 newuser@csunix ~/4301/12>
```

Figure 3. Commands to Compile, Assemble, Link, & Run Lab 12 (Part 2 of 2)