

1 (3 points)

2 Do not write a program for this exercise. Instead, for the program shown below merely draw an illustration
3 similar to the one included at the end of this document. Use the same *startup* stack frame shown there but
4 assume the following data type sizes and make any changes required by these assumptions:

5
6 type **long** is 4 bytes,
7 all other stack items are 2 bytes,
8

9 Do not show return values or library function calls on the stack. Your illustration may be drawn by hand as
10 long as it is neat, evenly arranged, and easily readable, although using Excel or making a table in Word is
11 easy. This is a paper exercise and should not be compared to any values obtained from actually running the
12 program.

```

14 int main(void) ← "startup function" calls main
15 {
16     int z = 3;
17     GetReady();
18     printf("Return from main");
19     return(EXIT_SUCCESS);
20 }
21
22 void GetReady(void) ←
23 {
24     long result;
25
26     gcd(128L, 96L);
27     printf("Return from GetReady");
28     return;
29 }
30
31 long gcd(long x, long y) ←
32 {
33     if (y == 0)
34         return(x);          /* see below */
35     return(gcd(y, x % y));  /* see below */
36 }
37
38
39
40
41

```

Function <i>main</i> (At Text Memory Address 0x1F0)	
Operation	Instruction Address
<code>z = 3</code>	0x1F0
Call to <i>GetReady</i>	0x200
Call to <i>printf</i>	0x220
return	0x240

Function <i>GetReady</i> (At Text Memory Address 0x108)	
Operation	Instruction Address
Call to <i>gcd</i>	0x108
Call to <i>printf</i>	0x116
return	0x12A

Function <i>gcd</i> (At Text Memory Address 0x79C)	
Operation	Instruction Address
the if statement	0x79C
reference to <i>x</i> (in first return)	0x7A0
first return	0x7A4
call to <i>gcd</i> (in last return)	0x7B2
last return	0x7C0

42 For Your Information (but not necessary for this exercise)

43 Anytime a function returns a value it does so by first storing that value into a "return location", which is possibly
44 but not necessarily in the previous stack frame (do not show this), then simply returning. The expression that
45 originally called the function may then access that "return location" to obtain the stored value if desired. Thus, a
46 statement such as "**return**(*x*);" can be interpreted (although never actually coded) as simply:

```

47 *returnPtr = x;          /* x's value gets placed in the "return location". */
48 return;                /* Whatever is currently in the "return location" remains there. */

```

51 and a statement such as "**return**(*gcd*(*y*, *x* % *y*);" can be interpreted (although never actually coded) as simply:

```

52 *returnPtr = gcd(y, x % y); /* gcd's return value gets placed in the "return location". */
53 return;                    /* Whatever is currently in the "return location" remains there. */
54

```

Illustration of the required format for this assignment

Taken directly from "Stack Activity During Program Run" in Note 12.5B of "Advanced C/C++ Notes"

BSS Variable: nextValue		<u>Stack Activity During Program Run</u>		Stack	
		Memory Addresses	Memory Addresses	Values	Description
Relative	Absolute	Relative	Absolute	Values	Description
Startup Stack Frame	??	??	??	??	??
BP=?? SP=0xFA7 PC=??	??	0xFA7	??	??	??
Stack Frame for <i>main</i>	BP+3	0xFA4	??	??	<i>function return address</i>
BP=0xFA1 SP=0xF9F PC=0x200	BP	0xFA1	??	??	<i>previous frame address</i>
Stack Frame for <i>GetReady</i>	BP-2	0xF9F	3	x	x
BP=0xF99 SP=0xF99 PC=0x108	BP+3	0xF9C	0x220	0x220	<i>function return address</i>
Stack Frame for 1 st call to <i>Recurse</i>	BP	0xF99	0xFA1	0xFA1	<i>previous frame address</i>
BP=0xF91 SP=0xF91 PC=0x79C	BP+6	0xF97	397	397	value
Stack Frame for 2 nd call to <i>Recurse</i>	BP+3	0xF94	0x116	0x116	<i>function return address</i>
BP=0xF89 SP=0xF89 PC=0x79C	BP	0xF91	0xF99	0xF99	<i>previous frame address</i>
Stack Frame for 3 rd call to <i>Recurse</i>	BP+6	0xF8F	39	39	value
BP=0xF81 SP=0xF81 PC=0x79C	BP+3	0xF8C	0x7BE	0x7BE	<i>function return address</i>
Stack Frame for 4 th call to <i>Recurse</i>	BP	0xF89	0xF91	0xF91	<i>previous frame address</i>
BP=0xF81 SP=0xF81 PC=0x79C	BP+6	0xF87	3	3	value
Stack Frame for 5 th call to <i>Recurse</i>	BP+3	0xF84	0x7BE	0x7BE	<i>function return address</i>
BP=0xF81 SP=0xF81 PC=0x79C	BP	0xF81	0xF89	0xF89	<i>previous frame address</i>