

NAME: _____

1. (20 points) The C code for the “isGreater” function from project 1 is as follows:

```

1. int isGreater(char *s1, char *s2) {
2.     while(*s1) {
3.         if ((*s1) > (*s2)) return 1;
4.         if ((*s1) < (*s2)) return -1;
5.         s1++; s2++;
6.     }
7.     if ((*s2) == 0) return 0;
8.     return -1;
9. }

```

For each of the following lines of x86 assembler code, identify the line number (or line numbers) of the C code which generated the x86 assembler code. Note – the x86 “movzbl” instruction copies from an 8 bit byte to a 32 bit long word, padding to the left with zeroes. The x86 “test” instruction performs a bitwise AND of its arguments, but does not save the results – just sets the condition codes.

A. _____

```

movl    8(%ebp), %eax
movzbl  (%eax), %edx
movl    12(%ebp), %eax
movzbl  (%eax), %eax
cmpb   %al, %dl
jge    .L27
movl    $-1, %eax
jmp    .L26
.L27

```

B. _____

```

jmp    .L24
.L28
...
.L24:
movl    8(%ebp), %eax
movzbl  (%eax), %eax
testb   %al, %al
jne    .L28

```

C. _____

```

addl    $1, 8(%ebp)
addl    $1, 12(%ebp)

```

D. _____

```

movl    12(%ebp), %eax
movzbl  (%eax), %eax
testb   %al, %al
jne    .L29
movl    $0, %eax
jmp    .L26
.L29

```

2. (20 points) The x86 instructions that enter and exit the “isGreater” function is as follows:

```
isGreater:
    pushl    %ebp
    movl    %esp, %ebp
    ...
.L26:
    popl    %ebp
    ret
```

- a. How many bytes are required for the “isGreater” stack frame? _____
 - b. If the “isGreater” function had any local variables, would the entry and exit code be different? If so, what other instructions would be required?
 - c. If the “isGreater” function invoked any lower level functions which required argument values, would the entry and exit code be different? If so, what other instructions would be required?
 - d. Would the “isGreater” function still work if we removed the `pushl %ebp; movl %esp,%ebp;` and `popl %ebp` instructions?
3. (10 points) Mark each statement below as either true (T) or false (F). Assume you are running GDB on the `l1st` binary that includes the “isGreater” function, as defined above, with the associated x86 assembler code as defined in problems 1 and 2.
- a. _____ If there is a breakpoint set at “isGreater”, then GDB will open a prompt after the execution of the “`movl %esp,%ebp`” instruction.
 - b. _____ If GDB provides a prompt after executing the “`move %esp,%ebp`” instruction at the top of the “isGreater” function, you can type the command “`x /3xw $ebp`” to see 1) the caller’s `%ebp`, 2) the return address to the caller, and 3) the first parameter to `isGreater`.
 - c. _____ If GDB provides a prompt at the `.L24` label (as in problem 1.b above), the GDB “next” instruction would cause GDB to open a new prompt after executing the instruction “`movl 8(%ebp),%eax`”
 - d. _____ If GDB provides a prompt at the `.L24` label (as in problem 1.b above), and `%ebp+8` has the value `0xffffdac0`, then the GDB command “`x /s 0xffffdac0`” will print the value of the `s1` string.
 - e. _____ If GDB provides a prompt at the `.L24` label (as in problem 1.b above), the GDB “`info isGreater`” command will print the author of the `isGreater` function.

4. (20 points) Consider the following Stack Information:

Address	Hex Value	Comments
0xff968a28	0xff968aa8	
0xff968a24	0x00000000	
...		
0xff9689e4	0x08049e00	
0xff9689e0	0x73007369	"is"
0xff9689dc	0x00000001	
...		
0xff9689cc	0x0804869d	
0xff9689c8	0xff968a28	
0xff9689c4	0x00000000	
...		
0xff968994	0x09992018	-> "this"
0xff968990	0xff9689e0	
0xff96898c	0x08048806	
0xff968988	0xff9689c8	%esp, %ebp

a. Identify the starting and ending addresses of each stack frame in this stack

Top of Frame	Bottom of Frame

b. When the currently executing function returns to its caller, what address will it return to?

c. The ASCII value for 'i' is 0x69 and the ASCII value for 's' is 0x73. Was the above stack created from a big-endian or a little-endian machine?

d. If the current function above is the "isGreater" function, what are the two parameters passed into the current invocation of that function?

e. If the above represents main calling insertWord calling isGreater, then which function has a local variable with the value "is"?

5. (30 points) Write a C function to insert a node into a binary tree. A node of the binary tree can be described by the following structure:

```
struct tnode {
    struct tnode * parent;
    int value;
    struct tnode * left;
    struct tnode * right;
};
```

You may use a function whose prototype is “struct tnode * makeTnode(int val)” which allocates space for a new instance of struct tnode, initializes the value field to the value of the val parameter, initializes all pointers to NULL, and returns a pointer to the new instance.

Your C function should have the prototype “void insertTree(int val, struct tnode *root)”, where val is an integer whose value is NOT already in the tree, and root is a pointer to the root node of an existing, non-empty binary tree. The tree has the property that the value of every node in the left sub-tree of any node is smaller than the value in the node itself, and the value in every node in the right sub-tree of a node is larger than the value in the node itself. The root’s parent pointer should be NULL, but every other node’s parent pointer should point at its parent node. Your function should add a new node to the tree for the “val” input, and maintain the ordering of the left and right sub-trees.

Hint: This function is simple if you write it recursively.