

Full Name: _____

'On my honor as a University
of Colorado at Boulder student
I have neither given nor received
unauthorized assistance on this
work.'

CSCI 2400, Fall 2014

Solutions to Practice Problems for Midterm #1

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name on the front. Put your name or student ID on each page.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- This exam is OPEN BOOK and you can use a *single page* of notes. Please attach your single sheet of notes to your exam when you're done. You can not use a computer or calculator. Good luck!

Problem	Page	Possible	Score
1	1	0	
2	2	0	
3	3	0	
Total		0	

1. [0 Points]

Assume we are running code on a 8-bit machine using two's complement arithmetic for signed integers. Also assume that T_{Max} is the maximum integer, T_{Min} is the minimum integer. Fill in the empty boxes in the table below. The following definitions are used in the table:

```
int y = -12
int x = 22;
```

Note: You need not fill in entries marked with “–”.

Each blank space is 1 point.

In the column labeled “Over/Under”, you should indicate if an overflow (carry out of the highest bit) or underflow (borrow from the highest bit) occurred.

Expression	Decimal Representation	Hex Representation	Over/Under?
–	31	0x1f	–
–	-3	0xfd	–
y	-12	0xf4	–
x+y	10	0xA	No
x + T_{Max}	-107	0x95	Yes
T_{Min} -x	106	0x6a	Yes

Given the following code

Given this C code	is translated to
<pre> struct node { int x; struct node *left, *right; }; int sum(struct node *p) { if (p) { return sum(p -> left) + sum(p -> right) + p->x; } else { return 0; } } </pre>	<pre> sum: pushl %ebp movl %esp, %ebp subl \$12, %esp movl %ebx, -8(%ebp) movl %esi, -4(%ebp) movl 8(%ebp), %ebx movl \$0, %eax testl %ebx, %ebx je .L4 movl 4(%ebx), %eax movl %eax, (%esp) call sum movl %eax, %esi movl 8(%ebx), %eax movl %eax, (%esp) call sum leal (%esi,%eax), %eax addl (%ebx), %eax .L4: movl -8(%ebp), %ebx movl -4(%ebp), %esi movl %ebp, %esp popl %ebp ret </pre>

Draw a picture of the stack frame as it would be at the point of the “je” instruction in the program, including any arguments passed to the function. Label each word in the stack frame and explain what each word is used for. If a word is used for multiple things, mention both. If it is not used, mention that.

	p	argument (“p”) from caller
	R/A	return address from caller
BEP →	oldbep	Saved BEP from caller
	oldesi	Saved ESI from caller
	oldebpx	Saved EBX from caller
ESP →	arg	Space for arugment for recursive calls to “sum”

3. [0 Points] Consider the following code for a C loop

Translate this code	into C
<pre> 00000000 <foo>: 0: pushl %ebp 1: movl %esp,%ebp 3: subl \$0x14,%esp 6: movl 0x10(%ebp),%eax 9: movw %ax,-0x14(%ebp) d: movl \$0x1,-0x4(%ebp) 14: movl 0xc(%ebp),%eax 17: subl \$0x18,%eax 1a: movl %eax,%edx 1c: movl 0x8(%ebp),%eax 1f: movb %dl,(%eax) 21: jmp 39 <foo+0x39> 23: movl -0x4(%ebp),%eax 26: addl 0x8(%ebp),%eax 29: movzwl -0x14(%ebp),%edx 2d: shll \$0x4,%edx 30: addl \$0x7,%edx 33: movb %dl,(%eax) 35: addl \$0x1,-0x4(%ebp) 39: movl -0x4(%ebp),%eax 3c: cmpl 0xc(%ebp),%eax 3f: jle 23 <foo+0x23> 41: movl 0xc(%ebp),%eax 44: addl 0x8(%ebp),%eax 47: movzbl (%eax),%eax 4a: leavel 4b: retl </pre>	<pre> char foo(char* x, int y, unsigned short z){ int i = 1; x[0] = y - 24; while (i <= y){ x[i] = 16*z + 7; i++; } return x[y]; } </pre>

You need to indicate the data types (short, int, char, *etc* as well as *unsigned* and **/&* if appropriate) for 'x', 'y', 'z', 'i', and the return type, fill out the rest of the blanks, and circle the correct operator in the conditional part of the for loop. If you need to use more space, use the back of the page – don't put in lots of circles and arrows in the template that makes it hard to grade. You should be able to represent your program using the template on right hand side of the table above; if you feel you can't, you can use the back of the page, but you're strongly advised to use that template as a guide to the structure of your program.

%ax	Lowest Two Bytes of %eax	%al	Lowest Byte of %eax
%dx	Lowest Two Bytes of %edx	%dl	Lowest Byte of %edx
movzwl	Move zero-extended word to double word.	movswl	Move sign-extended word to double word.
movzbl	Move zero-extended byte to double word.	movsbl	Move sign-extended byte to double word.
movzbw	Move zero-extended byte to word.	movsbw	Move sign-extended byte to word.
shl	Logical shift <i>dest</i> left by <i>src</i> bits.	shr	Logical shift <i>dest</i> right by <i>src</i> bits.
sal	Arithmetic shift <i>dest</i> left by <i>src</i> bits.	sar	Arithmetic shift <i>dest</i> right by <i>src</i> bits.