

Due: 9AM, Friday, October 26 in class

10 points total

Please show your answers (and all work) on these three sheets. Fold your paper the long way and write your name and Homework 6 on the outside.

1. (5 points) This problem tests your understanding of stack frames. It is based on the following C function:

```
#include <stdio.h>
char buf3[512];
int temp3;
int temp4;

int proc(int a, int b, int c)
{
    int temp2;
    int temp1;
    char buf1[2048];
    char buf2[1024];
    temp1 = 7*c + (a<<4);
    gets(buf1);
    temp2 = 11*temp1 + c*31;
    temp3 = 64* (c + 4* temp1 + 8*temp2);
    gets(buf2);
    gets(buf3);
    temp4 = (15* temp2) + (temp1<<5);
    printf("%s %s\n",buf1,buf2);
    return 12*temp1+11*temp2+10*temp3+9*temp4;
}
```

This yields the following machine code:

Dump of assembler code for function proc:	(continued from below left)
0x080483f0 <proc+0>: push %ebp	0x08048450 <proc+96>: call 0x804830c <gets@plt>
0x080483f1 <proc+1>: mov %esp,%ebp	0x08048455 <proc+101>: mov %esi,%eax
0x080483f3 <proc+3>: push %edi	0x08048457 <proc+103>: mov %ebx,%edx
0x080483f4 <proc+4>: push %esi	0x08048459 <proc+105>: shl \$0x4,%eax
0x080483f5 <proc+5>: push %ebx	0x0804845c <proc+108>: shl \$0x5,%edx
0x080483f6 <proc+6>: sub \$0xc0c,%esp	0x0804845f <proc+111>: sub %esi,%eax
0x080483fc <proc+12>: mov 0x10(%ebp),%edi	0x08048461 <proc+113>: add %edx,%eax
0x080483ff <proc+15>: mov 0x8(%ebp),%eax	0x08048463 <proc+115>: mov %eax,0x804a244
0x08048402 <proc+18>: lea 0x0(,%edi,8),%ebx	0x08048468 <proc+120>: lea 0xfffff3f4(%ebp),%eax
0x08048409 <proc+25>: sub %edi,%ebx	0x0804846e <proc+126>: mov %edi,0x8(%esp)
0x0804840b <proc+27>: shl \$0x4,%eax	0x08048472 <proc+130>: lea (%ebx,%ebx,2),%ebx
0x0804840e <proc+30>: add %eax,%ebx	0x08048475 <proc+133>: mov %eax,0x4(%esp)
0x08048410 <proc+32>: lea 0xfffff3f4(%ebp),%eax	0x08048479 <proc+137>: movl \$0x80485c0,(%esp)
0x08048416 <proc+38>: mov %eax,(%esp)	0x08048480 <proc+144>: call 0x804832c <printf@plt>
0x08048419 <proc+41>: lea (%ebx,%ebx,4),%esi	0x08048485 <proc+149>: mov 0x804a244,%ecx
0x0804841c <proc+44>: call 0x804830c <gets@plt>	0x0804848b <proc+155>: lea (%esi,%esi,4),%eax
0x08048421 <proc+49>: mov %edi,%eax	0x0804848e <proc+158>: mov 0x804a240,%edx
0x08048423 <proc+51>: shl \$0x5,%eax	0x08048494 <proc+164>: lea (%esi,%eax,2),%eax
0x08048426 <proc+54>: sub %edi,%eax	0x08048497 <proc+167>: add \$0xc0c,%esp
0x08048428 <proc+56>: lea (%ebx,%esi,2),%esi	0x0804849d <proc+173>: lea (%ecx,%ecx,8),%ecx
0x0804842b <proc+59>: add %eax,%esi	0x080484a0 <proc+176>: lea (%edx,%edx,4),%edx
0x0804842d <proc+61>: lea (%ebx,%esi,2),%eax	0x080484a3 <proc+179>: lea (%ecx,%edx,2),%edx
0x08048430 <proc+64>: lea (%edi,%eax,4),%eax	0x080484a6 <proc+182>: lea (%edx,%ebx,4),%ebx
0x08048433 <proc+67>: shl \$0x6,%eax	0x080484a9 <proc+185>: add %ebx,%eax
0x08048436 <proc+70>: lea 0xfffffbf4(%ebp),%edi	0x080484ab <proc+187>: pop %ebx
0x0804843c <proc+76>: mov %edi,(%esp)	0x080484ac <proc+188>: pop %esi
0x0804843f <proc+79>: mov %eax,0x804a240	0x080484ad <proc+189>: pop %edi
0x08048444 <proc+84>: call 0x804830c <gets@plt>	0x080484ae <proc+190>: pop %ebp
0x08048449 <proc+89>: movl \$0x804a040,(%esp)	0x080484af <proc+191>: ret

(continued above right) End of assembler dump.

Give the location of each of the following program variables. If the variable is located on the stack, give the location as an offset from %ebp, such as 12(%ebp) or %ebp + 12. If the variable is not located on the stack, then state where the variable is located.

temp1	_____	temp4	_____	buf1	_____
temp2	_____			buf2	_____

2. (5 points) The disassembly on the right was produced by `gdb` from the source code shown on the left.

```

/* CS 304 HW6, problem 2 */
#include <stdio.h>

int sum( int a[] )
{
    return a[0] + a[1] + a[2] + a[3];
}

int main(void)
{
    int u[4] = { 44, 55, 66, 77};
    int v = sum(u);
    printf("%d\n",v);
    return 0;
}

```

```

(gdb) disas sum+0 main+83
Dump of assembler code from 0x80483c0 to 0x8048433:
0x080483c0 <sum+0>:   push   %ebp
0x080483c1 <sum+1>:   mov    %esp,%ebp
0x080483c3 <sum+3>:   mov    0x8(%ebp),%edx
0x080483c6 <sum+6>:   pop    %ebp
0x080483c7 <sum+7>:   mov    0x4(%edx),%eax
0x080483ca <sum+10>:  mov    0xc(%edx),%ecx
0x080483cd <sum+13>:  add   (%edx),%eax
0x080483cf <sum+15>:  add   0x8(%edx),%ecx
0x080483d2 <sum+18>:  add   %ecx,%eax
0x080483d4 <sum+20>:  ret
0x080483d5 <sum+21>:  lea   0x0(%esi),%esi
0x080483d9 <sum+25>:  lea   0x0(%edi),%edi
0x080483e0 <main+0>:  lea   0x4(%esp),%ecx
0x080483e4 <main+4>:  and   $0xffffffff0,%esp
0x080483e7 <main+7>:  pushl 0xffffffffc(%ecx)
0x080483ea <main+10>: push   %ebp
0x080483eb <main+11>: mov    %esp,%ebp
0x080483ed <main+13>: push   %ecx
0x080483ee <main+14>: sub   $0x24,%esp
0x080483f1 <main+17>: lea   0xfffffec(%ebp),%eax
0x080483f4 <main+20>: movl  $0x2c,0xfffffec(%ebp)
0x080483fb <main+27>: movl  $0x37,0xfffff0(%ebp)
0x08048402 <main+34>: movl  $0x42,0xfffff4(%ebp)
0x08048409 <main+41>: movl  $0x4d,0xfffff8(%ebp)
0x08048410 <main+48>: mov   %eax,(%esp)
0x08048413 <main+51>: call  0x80483c0 <sum>
0x08048418 <main+56>: movl  $0x8048510,(%esp)
0x0804841f <main+63>: mov   %eax,0x4(%esp)
0x08048423 <main+67>: call  0x80482f8 <printf@plt>
0x08048428 <main+72>: add   $0x24,%esp
0x0804842b <main+75>: xor   %eax,%eax
0x0804842d <main+77>: pop   %ecx
0x0804842e <main+78>: pop   %ebp
0x0804842f <main+79>: lea   0xffffffffc(%ecx),%esp
0x08048432 <main+82>: ret
End of assembler dump.

```

Suppose that the registers have the following contents on entry to `main` at `0x080483e0`:

register	contents
<code>eax</code>	<code>0xbfc08734</code>
<code>ebx</code>	<code>0x4017fff4</code>
<code>ecx</code>	<code>0xfbf3b125</code>
<code>edx</code>	<code>0x00000001</code>
<code>esp</code>	<code>0xbfc086ac</code>
<code>ebp</code>	<code>0xbfc08708</code>
<code>esi</code>	<code>0x4001bca0</code>
<code>edi</code>	<code>0x00000000</code>
<code>eip</code>	<code>0x080483e0</code>

On the backside of the previous page, sketch the stack frames for this program at the point **just after** execution of the `mov` instruction in `sum` at location `0x080483c1` (`<sum+1>`). Your sketch should have the general appearance and level of detail of the stackframe sketches for the `swap_add`, `count23`, and `fib_rec` programs of Section 3.7 in the handout notes. All stack addresses and stack contents must be displayed in hexadecimal.

The program was compiled with the `-O2` flag for optimization.