

CIS 2107  
Computer Systems and Low-Level Programming  
Fall 2009  
Final

December 15, 2009

Name: \_\_\_\_\_

Page	Points	Score
1	20	
2	11	
3	6	
4	11	
5	17	
6	15	
7	4	
8	10	
9	11	
10	5	
11	15	
12	15	
Total:	140	

## Instructions

The exam is closed book, closed notes. You may *not* use a calculator, cell phone, etc.

If the question reads, “answer briefly”, it means just that. A sentence or two at most should be sufficient.

Unless otherwise specified, you *may* use functions from the Standard C Library.

There is some extra room on the back page.

Good luck.

(1 point) 1. What would I type in the shell to change to the parent of my current directory?

1. \_\_\_\_\_

(1 point) 2. What would I type to run the program `prog` taking its input from `in.txt` and writing its output to `out.txt`?

\_\_\_\_\_

(1 point) 3. For the memory allocator program, you put your `myMalloc( )` and `myFree( )` functions into a file called `myMalloc.c`. This file doesn't contain a `main( )`. What's the command that you'd type to instruct `gcc` to compile this file without running the linking process, so that you'd get the file `myMalloc.o`?

\_\_\_\_\_

4. When I type the command `gcc -o prog file1.c file2.c file3.c`, there are several steps which occur during the creation of the executable file `prog`. Among these are preprocessing, compiling, assembling, and linking. Very briefly describe what happens during each of these steps.

(2 points) (a) preprocessing

\_\_\_\_\_  
\_\_\_\_\_

(2 points) (b) compiling

\_\_\_\_\_  
\_\_\_\_\_

(2 points) (c) assembling

\_\_\_\_\_  
\_\_\_\_\_

(2 points) (d) linking

\_\_\_\_\_  
\_\_\_\_\_

5. Some C keywords. Provide a very brief description of what each of the following keywords mean in the contexts presented.

(2 points) (a) A variable inside a function declared as `static`

\_\_\_\_\_

(2 points) (b) A variable outside a function declared as `static`

\_\_\_\_\_

(2 points) (c) A variable outside a function declared as `extern`

\_\_\_\_\_

(3 points) 6. What is a *system call*? In your very brief explanation, please be sure to mention how they're different

from basic function calls.

---

---

---

---

- (2 points) 7. Very briefly, when we declare a struct inside a .h file in C, why is it that we enclose it inside `#ifndef`, `#define`, ..., `#endif`?

---

---

---

---

- (3 points) 8. If I compile a C program on lucas, explain why the binary won't run on my Windows computer at home or on an old Mac? Please do not write, "because they're different operating systems". Be more specific about what the major problems are.

---

---

---

---

9. some conversions

- (1 point) (a) 160 gbytes = ? mbits

(a) \_\_\_\_\_

- (1 point) (b) 24 bytes = ? gbits

(b) \_\_\_\_\_

- (1 point) (c) 24 tbytes = ? kbytes

(c) \_\_\_\_\_

10. Using the approximation trick that we talked about in class, about how much are each of the following?

- (1 point) (a)  $2^{15}$

(a) \_\_\_\_\_

- (1 point) (b)  $2^{36}$

(b) \_\_\_\_\_

- (1 point) (c)  $2^{48}$

(c) \_\_\_\_\_

(1 point) 11. Convert  $206_{10}$  to base 2.

(1 point) 12. Convert from base 10 to base 16:  $211_{10}$

(2 points) 13. What is  $10110011_2 + 11101_2$  in base 2?

$$\begin{array}{r} 1011\ 0011_2 \\ + \quad 1\ 1101_2 \\ \hline \end{array}$$

(2 points) 14. What is  $6CB81_{16} + BDF_{16}$  in base 16?

$$\begin{array}{r} 6CB81_{16} \\ + \quad BDF_{16} \\ \hline \end{array}$$

15. **Some bit operations.** If we have `char i = 0x7C`, `j = 0x31`;, what is the result of the following operations? Your answer must be in the form of exactly two hex digits<sup>1</sup>.

(1 point) (a) `~i`

(a) \_\_\_\_\_

(1 point) (b) `!!i`

(b) \_\_\_\_\_

(1 point) (c) `i^j^j`

(c) \_\_\_\_\_

(1 point) (d) `i||j`

(d) \_\_\_\_\_

(1 point) (e) `i>>2`

(e) \_\_\_\_\_

16. What is printed by the following code, assuming that the address of `s` is 1000, and the address of `A` is 1150? You may assume that the size of `ints` is the standard for a 32-bit machine. (Note: it is not a misprint that `int *p` points to `s` and `*q` points to `A[ ]`.)

```
...
char s[25];
...
int A[10];

int *p=(int*)&s[0];
char *q=(char*)&A[0];

printf("address of (p+10): %p\n", (void*)(p+10));
printf("address of (q+10): %p\n", (void*)(q+10));
...
```

(3 points) (a) address of `(p+10)`:

(a) \_\_\_\_\_

(3 points) (b) address of `(q+10)`:

(b) \_\_\_\_\_

---

<sup>1</sup>Yes. In real life, some of these operations could involve promoting the operands to 32-bit ints. Ignore that for now. Just pretend that we're living in the land of 8-bit arithmetic.

(12 points) 17. REDACTED MULTI-PART QUESTION ABOUT TWO'S COMPLEMENT ARITHMETIC.

(5 points) 18. How would the number  $-181.40625_{10}$  be stored in a C `float` variable?

(5 points) 19. For each item in the following piece of code, indicate in which memory segment the item will be stored:

```
1 char *s = "This exam is kind of long.";
2 int x;
3
4 char *func(void)
5 {
6     char *p;
7     x=10;
8     p=malloc(x);
9     return p;
10 }
```

(5 points) 20. Without using the mathematical operators  $\{+, -, *, /, \%\}$ , write a function `div8` which is passed an `int x`, and returns 1 if `x` is evenly divisible by 8 or 0 otherwise.

(5 points) 21. Write a function called `isBigEndian( )`, which returns 1 if the machine you're running on is a big-endian

machine or 0 if it's a little-endian machine.

22. Some tricky declarations. Write a very brief description in English of what is declared. For example, if the question is `int func(int A[])`, you'd write, "func is a function which is passed an array of int and returns an int".

(1 point) (a) `int (*j)[10]`

---

---

(1 point) (b) `void (*j)()`

---

---

(1 point) (c) `char *(*j)(int[])`

---

---

(1 point) (d) `int (*j[])(char*);`

---

---

23. What is the value of each of the following after `func( )` is called?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct {
5      int *p;
6      int A[3];
7      int x;
8  } Stuff;
9
10 void func(int[], int*, int*, Stuff, Stuff*);
11
12 int main(void)
13 {
14     Stuff s1, s2;
15
16     int i=0, j=10;
17     int A[3] = {20,30,40};
18     s1.p=&i;
19     s1.A[0]=50;
20     s1.x=70;
```



```

21     s2=s1;
22     func(A, &i, &j, s1, &s2);
23     return 0;
24 }
25
26 void func(int A[], int *p, int *q, Stuff s1, Stuff s2)
27 {
28     A[0]=555;
29     *p=666;
30     q=p;
31     *q=777;
32     *(s1.p)=888;
33     s1.A[0]=999;
34     s1.x=1111;
35     s2 = (Stuff*)calloc(1, sizeof(Stuff));
36     *s2.x=3333;
37     s2->A[0]=4444;
38 }
39

```

(1 point) (a) i

(a) \_\_\_\_\_

(1 point) (b) j

(b) \_\_\_\_\_

(1 point) (c) A[0]

(c) \_\_\_\_\_

(1 point) (d) \*(s1.p)

(d) \_\_\_\_\_

(1 point) (e) s1.x

(e) \_\_\_\_\_

(1 point) (f) s1.A[0]

(f) \_\_\_\_\_

(1 point) (g) s2.A[0]

(g) \_\_\_\_\_

(3 points) 24. What's the problem with this function?

```

1  #include <ctype.h>
2
3  char *lower(char *s)
4  {
5  #define LEN 25
6      char low[LEN];
7      int i=0;
8
9      while (i<LEN-1 && *s!='\0') {
10         low[i]=tolower(*s);
11         s++;
12         i++;
13     }
14     low[i]='\0';
15     return low;
16 }

```

\_\_\_\_\_

\_\_\_\_\_

- (5 points) 25. For each of the following, suppose that `%eax` contains the value  $x$ , `%ecx` contains  $y$ . What's stored in `%edx` after the each operation?

expression	result
<code>leal 0xC(%eax), %edx</code>	
<code>leal (%eax,%ecx), %edx</code>	
<code>leal (%eax,%ecx, 4), %edx</code>	
<code>leal 5(%eax,%eax,8), %edx</code>	
<code>leal 0xA(,%ecx,8), %edx</code>	

26. Assuming that we compile and run the following function on a Linux box on a 32-bit Intel chip, write an assembly expression indicating where we'd find each of the following:

```

1  int func(int x, int y)
2  {
3      int l;
4      char s[10];
5
6      return x+y;
7  }
```

(1 point) (a) `x`

(a) \_\_\_\_\_

(1 point) (b) `y`

(b) \_\_\_\_\_

(1 point) (c) `l`

(c) \_\_\_\_\_

(1 point) (d) `s`

(d) \_\_\_\_\_

(1 point) (e) the return address

(e) \_\_\_\_\_

(1 point) (f) where the calling function can find the value returned by `func( )`

(f) \_\_\_\_\_

(5 points) 27. Write the few lines of code which prints A, a  $n \times m$  array of int row-by-row using only a single loop.

(10 (bonus)) 28. Write a C function equivalent to the following assembly (no credit for an answer containing inline assembly):

```
1          .text
2          .type func, @function
3          .globl func
4  func:
5          pushl %ebp
6          movl %esp, %ebp
7          subl $24, %esp
8          movl 8(%ebp), %eax
9          cmpl 12(%ebp), %eax
10         jge L2
11         movl $1, %eax
12         jmp  L4
13  L2:
14         movl $0, %eax
15  L4:
16         movl %ebp, %esp
17         popl %ebp
18         ret
```

(5 points) 29. Sketch out the `myFree( )` function from the memory allocator assignment in pseudocode (or an outline in English).

(10 points) 30. Write a function which is passed a pathname. The function returns a new string which represents the file part of the path. For example, if the path passed is `"/usr/local/bin/junk.txt"`, the function returns a pointer to a new string containing `"junk.txt"`. If the last character in the path is a `"/"`, the function returns `NULL`. If necessary, the caller is responsible for freeing the memory used by the returned string.

- (15 points) 31. Write a program in which a series of one or more filenames will be passed at the command line. Of all of the words in all of the files, your program prints the word which would appear last in the dictionary ignoring case.

(extra space)