CIS 2107
Computer Systems and Low-Level Programming
Spring 2010
Midterm

March 16, 2010

Name: ________________________________________________

<table>
<thead>
<tr>
<th>Page</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

**Instructions**
The exam is closed book, closed notes. You may *not* use a calculator, cell phone, etc.
For the following questions, you can assume that my home directory is the `jfiore` directory.

1. Unix shell stuff.

(a) If I’m in my home directory, what’s the command to compile `prog.c` into the executable called `prog`?

Solution: `gcc -o prog prog.c`

(b) If I’m in my home directory, what’s the command to compile `prog.c` but not to run the linker (so we end up creating a file called `prog.o`)?

Solution: `gcc -c prog.c`

(c) What’s the command to run the program `prog` taking its input from `stuff.txt` and writing its output to `out.txt`?

Solution: `prog < stuff.txt > out.txt`

(d) If I’m in the `2107` directory, what’s the single command that I’d type in order to create a `03` directory in the `assign` directory?

Solution: `mkdir assign/03`

(e) If I’m in the `2107/assign/02` directory, what’s the one command that I’d type in order to delete `stuff.txt`?

Solution: `rm ../../../stuff.txt`

2. Why is it that when we declare a struct, functions, etc inside a .h file, we enclose it in a #ifndef #define #endif block?

Solution: So that the preprocessor does not ever include the definitions more than once.

3. (a) 152 bytes = ? kbytes

(a) 0.152
(1 point) (b) 72 tbits = ? bytes

(b) 9,000,000,000,000

(1 point) (c) 16 mbytes = ? bits

(c) 128,000,000

(1 point) (d) 5 secs = ? milliseconds

(d) 5,000

(1 point) (e) 72 mbits = ? gbytes

(e) 0.009

Convert each of the following from base 10 to base 2 and base 16

(1 point) 4. (a) 193

Solution:

193 = 96 * 2 + 1
96 = 48 * 2 + 0
48 = 24 * 2 + 0
24 = 12 * 2 + 0
12 = 6 * 2 + 0
6 = 3 * 2 + 0
3 = 1 * 2 + 1
1 = 0 * 2 + 1

193_{10} = 11000001_{2} = C_{16}

(b) 126

Solution:

126 = 63 * 2 + 0
63 = 31 * 2 + 1
31 = 15 * 2 + 1
15 = 7 * 2 + 1
7 = 3 * 2 + 1
3 = 1 * 2 + 1
1 = 0 * 2 + 1

126_{10} = 1111110_{2} = 7E_{16}

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

(1 point) (a) 2^{17}

(a) 128 thousand

points: _______ 2 of 10
out of a possible 7

question 5 continues...
(b) \[ 2^{45} \]

(b) \[ 32 \text{ trillion} \]

(c) \[ 2^{34} \]

(c) \[ 16 \text{ billion} \]

6. What is \(11101010_2 + 1001111_2\) in base 2?

\[
11101010_2 \\
+1001111_2 \\
10011100_2
\]

7. What is \(95F8_{16} + A398_{16}\) in base 16?

\[
95F8_{16} \\
+ A398_{16} \\
A031B_{16}
\]

8. In hex, what is the smallest integer that can be represented by a 32-bit two's complement int?

8. \[ 0x80000000 \]

9. In hex, what is the largest integer that can be represented by a 32-bit two's complement int?

9. \[ 0xFFFFFFFF \]

10. In hex, what is \(-1\) as a 32-bit two's complement int?

10. \[ 0xFFFFFFFF \]

11. What is printed by the following? (If you've forgotten, the %x formatting character for printf simply means to print out the value in hex.)

```c
int x = 0x01234567;
char c = (char)x;
printf("c=%x\n", c);
```

Solution: We get the last byte of x, so \(0x67\).

12. What is printed by the following?

```c
unsigned char x=255;
x+=3;
printf("x=%d\n", x);
```

12. \(x=2\)

13. Some bit operations. If we have \(\text{char } i = 0xA7, j = 0x6C;\), what is the result of the following operations? Your answer must be in the form of exactly two hex digits\(^1\).

\(^1\)Ignore the possibility of promotion to 32-bit ints. Behave as though we’re living in the land of 8-bit arithmetic.

points: \[ \text{out of a possible 16} \]
(1 point) (a) \texttt{\textasciitilde i}

\begin{center}
(a) \texttt{0x58}
\end{center}

(1 point) (b) \texttt{!i}

\begin{center}
(b) \texttt{0x00}
\end{center}

(1 point) (c) \texttt{!!i}

\begin{center}
(c) \texttt{0x01}
\end{center}

(1 point) (d) \texttt{i \& 0xFF}

\begin{center}
(d) \texttt{0xA7}
\end{center}

(1 point) (e) \texttt{i^j^j}

\begin{center}
(e) \texttt{0xA7}
\end{center}

(1 point) (f) \texttt{i|j}

\begin{center}
(f) \texttt{0x01}
\end{center}

(1 point) (g) \texttt{i<<1}

\begin{center}
(g) \texttt{0x4E}
\end{center}

(1 point) (h) \texttt{ij}

\begin{center}
(h) \texttt{0xEF}
\end{center}

points: \underline{4} of 10

out of a possible 8

exam continues...
(5 points) 14. What’s wrong with the following code?

```c
char *f(char t[ ])
{
    int i;
#define LEN 10
    char s[LEN];
    for (i=0; t[i]!="\0" && i<LEN-1; i++)
        s[i]=t[i];
    s[i]="\0";
    return s;
}
```

**Solution:** The preprocessor directive is just fine. The problem is that we’re returning a pointer to local memory.

15. **Two’s complement.** The function `TC8()` takes an 8-bit signed int and returns its two’s complement. What would be the value returned by applying `TC8()` to the following? Your answer should be in the form of two hex digits.

(2 points) (a) 0x58

(a) ___________

**Solution:** The details:

\[
0x58 = 01011000_2 \\
\sim 0x58 = 10100111_2 \\
\sim 0x58 + 1 = 10101000_2 \\
= 0xA8
\]

(2 points) (b) 0x91

(b) ___________

**Solution:** The details:

\[
0x91 = 10010001_2 \\
\sim 0x91 = 01101110_2 \\
\sim 0x91 + 1 = 01101111_2 \\
= 0x91
\]

points: ______ 5 of 10 exam continues...
0x91 = 10010001₂
\sim0x91 = 01101110₂
\sim0x91 + 1 = 01101111₂
= 0x6F

(4 points) 16. On my little-endian machine, ints are 4 bytes. If x=0x01234567 and &x=1000, how exactly is x stored? Your answer should be a list of addresses and what’s stored at each address.

Solution:

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>67</td>
</tr>
<tr>
<td>1001</td>
<td>45</td>
</tr>
<tr>
<td>1002</td>
<td>23</td>
</tr>
<tr>
<td>1003</td>
<td>01</td>
</tr>
</tbody>
</table>

17. If I have the following:

```c
int main(void)
{
    int a=10;
    int b=20;
    int *p=&a;
    int *q=p;
    char *cp=(char*)&a;
    
    ...
    p++;
    *q++;
    cp+=4;
}
```

and we suppose that on my particular machine, ints and pointers are 4-bytes and memory is laid out like this:

```
 a  1000
 b  1004
 p  1008
 q  1012
 cp 1016
```

Solution:

```
 a  1000 11
 b  1004 20
 p  1008 1004
 q  1012 1000
 cp 1016 1004
```

points: _______ 6 of 10 question 17 continues...
what do you see if you print:

(a) a
(b) &a
(c) b
(d) &b
(e) p
(f) *p
(g) &p
(h) q
(i) *q
(j) &q
(k) cp
(l) *cp
(m) &cp

18. What is the value of each of the following after func( ) is called?²

```c
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
typedef struct {
    int x;
    int A[1];
    int *p;
} Junk;
```

²In case you’ve forgotten, strcpy(char *d, char *s) copies the string s to the string d including the terminating null character.

points: ________ 7 of 10 question 18 continues...
CIS 2107
Computer Systems and Low-Level Programming
Midterm
March 16, 2010

```c
void func(char[], Junk, int*, int);

int main(int argc, char **argv)
{
    int n = 11;
    int *p = (int*)malloc(sizeof(int));
    char str[25];
    Junk j;
    strcpy(str, "You're mostly finished");
    j.x = 22;
    j.A[0] = 33;
    j.p = (int*)malloc(sizeof(int));
    *(j.p) = 44;
    *(j.p) = 999;
    *(j.p) = 999;
    *p = 55;
    func(str, j, p, n);
}

return 0;
}
```

(a) (2 points) n?
(a) 11

(b) (2 points) *p?
(b) 56

(c) (2 points) str=?
(c) YOU'RE MOSTLY FINISHED

(d) (2 points) j.x=?
(d) 22

(e) (2 points) j.A[0]=?
(e) 33

(f) (2 points) *(j.p)=?
(f) 999

(g) (2 points) *p=?
(g) 56

points: ________  out of a possible 14  8 of 10  exam continues...
19. (6 points) Write a function which takes as an argument, an int x. The function returns the number of 1s that would appear in x’s binary representation. In order to make your function portable, you should not assume that ints are a particular size (i.e., don’t assume that ints are 4 bytes).

Solution:

(8 points) 20. Implement the function replace( ), which takes exactly three arguments, a C string s, a character c, and a C string r. The function returns a pointer to a new string, which contains all of the characters of s, but with each instance of the character c replaced with the string r. For example, if s = “Fiore is a x lecturer and he dresses xly”, c = ‘x’, and r = “bad”, the returned string is “Fiore is a bad lecturer and he dresses badly”. The function is responsible for allocating enough memory to hold the returned string.

Solution:

```c
#include <stdlib.h>

int len(char*);
int num_instances(char*, char);
char *replace(char*, char, char*);

char *replace(char* s, char c, char *r)
{
    int lenret; /* length of the string to return */
    char *retstr; /* string to return */
    int i, j, k;

    lenret=len(s)+num_instances(s,c)*(len(r)-1)+1;
    if ((retstr = (char*)malloc(lenret))==NULL)
        return NULL;

    for (i=0, j=0; s[i]!='\0'; i++) {
        if (s[i]==c) {
            /* append r to retstr */
            for (k=0; k<len(r); k++, j++)
                retstr[j]=r[k];
        } else { /* just copy */
            retstr[j]=s[i];
            j++;
        }
    }

    return retstr;
}

int len(char *s)
{
    char *t=s;
    while (*t!='\0')
        t++; 
    return t-s;
}
```
(7 points) 21. Write a program which takes at the command line a list of files, and prints each of the files to the screen.

Solution: