CMSC 212 Midterm #2 (Fall 2005)

Name _________________________ Signature ________________________

Discussion Section Time (circle one):  12:00 1:00 2:00 3:00 4:00 5:00
Elena Sorelle Morgan

(1) This exam is closed book, closed notes, and closed neighbor. No calculators are permitted. Violation of any of these rules will be considered academic dishonesty.

(2) You have 75 minutes to complete this exam. If you finish early, you may turn in your exam at the front of the room and leave. However if you finish during the last ten minutes of the exam please remain seated until the end of the exam so you don't disturb others. Failure to follow this direction will result in points being deducted from your exam.

(3) Write all answers on the exam. If you need additional paper, we will provide it. Make sure your name is on any additional sheets.

(4) Partial credit will be given for most questions assuming we can figure out what you were doing.

(5) Please write neatly. Print your answers, if that will make your handwriting easier to read. If you write something, and wish to cross it out, simply put an X through it. Please clearly indicate if your answer continues onto another page.

(6) The CMSC 212 Final Exam is scheduled for Wednesday, December 15(4:00-6:00pm) Location TBA.

<table>
<thead>
<tr>
<th>Page</th>
<th>Possible</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1.) [25 points] Define and explain the following as requested:

a) Assuming that you want to dynamically link a library and use the functions described in that library into the function you are writing, describe the code that would need to appear within the current function so that the functions of that library could be called. Make sure you include all steps that would be used if only this function needs to access those library functions.

b) Write all of the pieces of code that would be necessary so that the one line \( z = \text{callit}(x, y) \) could be used to call the add function if the numbers are both positive or the multiply function if either (or both) of the integers are non-positive.

```c
int add(int a, int b){
    return a+b;
}

int multiply(int a, int b){
    return a*b;
}
```

/* write the code for any type definitions or variable declarations that would be needed also */

```c
int funct(int x, int y){
    int z;
    /* write the code that would go here so that z gets the correct value as described above when the line shown below is executed */
    z = callit(x, y)
}
```
c) Given that the following line is used to define the library flags in your make file:

\texttt{LIBFLAGS = -nostdlib -shared -fPIC -Wl,-soname,$@.1}

Describe the purpose of the $@ that appears near the end of the line. Tell when this would be useful.

d) Answer the following questions about using lcov.

1. What needs to appear in the Makefile so that lcov can be accessed?

2. What steps do you need to take at the shell prompt in order to use lcov?
e) Compare and contrast Whitebox testing and Blackbox testing. Compare and contrast means to specify both similarities and differences.

f) Write, in hex, the two’s complement representation of the 32-bit value “-9”.
2.) [23 points] Write the complete main that would exit setting status to -1 and with a printed error message if there are not at least two arguments (in addition to the application which is in argv[0]). If there are at least two arguments, it should count how many would be positive integers (containing only numeric digits) and how many must only be considered as strings. It should then print out both counts. It then also prints the sum of the positive integers.

You may use the functions:

   int isdigit(char);
   int atoi(const char *);

For example:

<table>
<thead>
<tr>
<th>call from shell prompt</th>
<th>output printed to the screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>countthem a b 12 4 c</td>
<td>Numeric: 2 non-Numeric: 3   Sum: 16</td>
</tr>
<tr>
<td>countthem</td>
<td>There are not at least two arguments</td>
</tr>
<tr>
<td>countthem 12 23 13 1324a</td>
<td>Numeric: 3 non-Numeric: 1   Sum: 48</td>
</tr>
</tbody>
</table>
3.) [15 points] Answer the following questions about multiple processes from within a single C source file.

a) When the fork function is called, what value does the child process have as the return value of that function?

b) When the fork function is called, what values could the parent process have as the return value of that function?

c) If the value of a variable named parentvar is set to 5 before the fork command is called and the following line appears in the child portion of the code, circle the letter of the statement that best describes what will happen.

```
printf("%d\n", parentvar);
```

a) The child will declare it’s own variable named parentvar with a random initial value.
b) The child will declare it’s own variable named parentvar with a 0 as the initial value.
c) The child will declare it’s own variable named parentvar with a 5 as the initial value.
d) There will be a compilation error because that variable is in the parent’s space.
e) There will be a runtime error as the child accesses the parent’s space.
f) None of the above.

d) If the child changes the value of parentvar to 7, after the fork but before the child finishes, select the line that best describes what will happen if the following line appears after the wait in the parent process (so this line is running after the child has completed). Assume the parent did nothing to modify the value of this variable since it was originally set before the fork in the previous question.

```
printf("%d\n", parentvar);
```

a) The parent will print the value 7 since they share the same variable space.
b) The parent will print the value 5 since the child changed its own copy of the variable.
c) The parent will print a random value since the space got corrupted -by the child’s attempt to change the value.
d) There will be a compilation error because they can not both have access to the same variable.
e) There will be a runtime error as the child accesses the parent’s space so this line won’t be executed at all.
f) None of the above.
4.) [21 points] Project 4 as defined used the following types. Write function called “distorttree” which will make a new tree that has all of the arithmetic operators changed to their opposite (plus becomes minus, minus becomes plus, mult becomes div and div becomes mult). The function returns the distorted tree through the return value of the function. You may write helper functions as needed. Names have been reduced to save some writing time.

typedef enum { operatorNode, variableNode, constantNode } nodeType;
typedef enum { plusOp, minusOp, multOp, divOp, equalOp } operatorType;

typedef struct _node {
    nodeType type;
    struct _node *left;
    struct _node *right;
    int value;
    operatorType operator;
    char *name;
} ASTnode;

ASTnode *createOperatorNode(operatorType op, ASTnode *left, ASTnode *right); /* assume these */
ASTnode *createConstantNode(int constant); /* functions have been written */
ASTnode *createVariableNode(char *name); /* and tested already for you */
ASTnode *copyTree(ASTnode *node); /* you do not need to write them*/

---------------------------------------------------------------------------
ASTnode *distortTree(ASTnode *node); /*This is the prototype for the function you must implement*/
5.) [16 Points] Recall the heap manager discussed in class, which allocates space using 8-byte chunks. Assume a total heap space of 64 bytes, and that the interface to the heap manager consists of 'malloc()' and 'free()' with the usual semantics. The shown heap does not contain the header discussed in class to manage the blocks – just the blocks requested are allocated. Assuming each square represents an 8-byte chunk, we can pictorially represent the state of the heap at some point as follows:

This shows the heap being broken into three blocks. The center of those three blocks is allocated and the two on the ends are free.

a) Assuming a **first-fit** allocation strategy, that the memory is completely unallocated before starting, and the sequence of malloc() and free() operations below, annotate the following diagram by shading allocated regions to indicate the state of the heap immediately after each line of the program.

```
p1 = malloc(15);
p2 = malloc(20);
p3 = malloc(16);
free(p2);
p4 = malloc(4);
free(p3);
```

b) Repeat the above assuming **best-fit**.

```
p1 = malloc(15);
p2 = malloc(20);
p3 = malloc(16);
free(p2);
p4 = malloc(4);
free(p3);
```

c) Define internal and external fragmentation.