NCSU ECE 209 Sections 602 and 604
UNCA CSCI 373 Section 002
Exam 2 Fall 2010
16 November, 2010

This is a closed book and closed notes exam. It is to be turned in by 12:05 pm. Calculators, PDA's, cell phones, and any other electronic or communication devices may not be used during this exam.

Please read and sign the following statement:

I have neither given not received unauthorized assistance on this test.

Name: _____

If you want partial credit for imperfect answers, explain the reason for your answer!

Problem 1 (5 points)
Complete the following section of C code so that it copies to first five letters of your name into the first five elements of the array `bunchChars` declared below. (Consider using `strncpy`.)

```
char bunchChars[1000] ;
```

Problem 2 (10 points)
Below is a sequence of C statements defining some variables

```
int    viii =   8 ;
double half = 0.5 ;
```

The table below contains two columns. The leftmost column is a C expression. In the rightmost column write the value of the expression as a C literal. If the value is a double, be sure to use the C syntax for writing doubles. Some of these are tricky, but none require complex arithmetic.

`8 * 0.5`	
`half * 0`	
`8/4*2`	
`5 < 6 <= 6`	
`2*(1/2)`	
`1 && 0 == 0`	
`half = 5`	
`viii && 4+4 == 8`	
`! viii == 0`	
`viii += 2`	

Problem 3 (6 points)
In the six boxes below are six possible values for C `doubles`. Cross out the ones that would **not** be legal literals for `double` values in C.

`3.3^5`	`eieio`	`1e-10`
`1000`	`.1e1`	`.16666666666666`

Problem 4 (9 points)
Given the following prototypes for C functions and declarations of C variables.

```
int f(int    m, int   n, int  *o) ;
void g(int  *p, int  *q) ;
void h(int **r, int   s) ;
int a ;
int *b ;
int c[15] ;
```

In the following list there are nine proposed calls of the declared functions which might be legal in C. If the call is not legal, explain why and indicate which of the arguments do not match the parameters of the prototype.

`f(c[3], 0.0, &b)`
`f(a, a)`
`g(&b, &b)`
`g(c, c)`
`g(&a, &a)`
`g(c+5, c+10)`
`h(&b, b)`
`h(b, &b)`
`h(&c[3], c[3])`

Problem 5 (20 points)
For the following block of C code, draws the pointers as they exist when the block is executed and write the output that will be printed when the block is executed. This would be a good place to explain your work and do some drawing.

```
int a, b ;
int *p, *q ;
int v[2] ;
a = 10 ;
b = 20 ;
for (a = 0; a<2; ++a) {
  v[a] = 30 + a*10 ;
}
printf("%d %d %d %d\n", a, b, v[0], v[1]) ;
```

```
p = &a ;
*p = b + 5 ;
v[0] = *p + 7 ;
printf("%d %d %d %d\n", a, b, v[0], v[1]) ;
```

```
p = &a ;
q = &b ;
*p = *q ;
*q = *p ;
printf("%d %d %d %d\n", a, b, v[0], v[1]) ;
```

```
q = &v[0] ;
p = q ;
b = *++q ;
a = ++*p ;
printf("%d %d %d %d\n", a, b, v[0], v[1]) ;
```

```
p = &v[0] ;
q = &v[1] ;
a = q-p ;
b = *q-*p ;
printf("%d %d %d %d\n", a, b, v[0], v[1]) ;
```

For the problems on this page assume the following declarations:

```
int dice[5] ;
int histo[6] ;
int i,j,m,s ;      /* for various stuff */
```

Problem 6 (8 points)
Write a small section of C code to set `m` to the largest of all the numbers stored in `dice`.

Problem 7 (3 points)
Suppose there is a function with the following prototype that adds the `n` elements stored in an integer array `V`.

```
int sum(const int *V, int n) ;
```

Write a C statement to set `s` to the sum of the elements in `dice`.

Problem 8 (3 points)
Now define a function prototype that would be appropriate for finding the number of times an integer variable `x` occurs within the first `n` elements of an array of integers `V`.

Problem 9 (8 points)
Now implement the function for which you wrote the prototype in Problem 8.

Problem 10 (3 points)
Use the function you defined in Problem 9 to set `m` to the number of times the value 2 occurs within the array `dice`.

In these problems you are to use the C character, string and file functions to solve a few problems. These functions are described in the distributed C Reference Card.

The following declarations (similar to those of the card) will be used in the problems.

```
char c ;                    /* a character */
char *s ;                   /* a string */
```

Keep in mind that "regular" strings can always be passed to functions accepting constant strings.

In all of these examples, an integer variable `yorn` is to be set to 1 if the condition is true. You may assume the following definition and initialization of `yorn`.

```
int yorn = 0 ;
```

Try not to use loops in your solution. Full credit will be given only for loop-free code. You are free to add any additional variable declarations that are required for your answers.

Problem 11 (3 points)
Set `yorn` to 1, if `c` is a digit or lowercase letter.

Problem 12 (3 points)
Set `yorn` to 1, if the character `'@'` appears in `s`.

Problem 13 (3 points)
Set `yorn` to 1, if `s` has at least one character *and* the last character in `s` is `'!'`.

Problem 14 (4 points)
Set `yorn` to 1, if the character `'@'` appears in `s` *and* the characters immediately following the *first* `'@'` in `s` are `"cs.unca.edu"`.

Problem 15 (4 points)
Set `yorn` to 1, if the file `read.me` can be opened for reading.

Problem 16 (8 points)
Set `yorn` to 1, if the file `read.me` can be opened for reading and the file consists of nothing more than the single letter `'?'`.