The University of Arizona
Department of Electrical and Computer Engineering

ECE275 Midterm Examination
Spring 2001

Instructor: Dr. Michael Marefat

Name: _____

Please Note:

1. **Total Exam Points = 100.**
2. **Maximum Allowed Time = 70 minutes.**
3. **Closed books, closed notes, closed neighbor.**
4. **Calculators not required.**
5. **Provide answers in the given spaces only.**

Problem 1 (20 points)

For each of the following questions indicate whether the statement is TRUE or FALSE: (2 pts. each)

True/False 1. The following code loops 3 times:

```
int t;
for (t = 100; t > 24; t = t / 2)
{
     printf("%d\n",t);
}
```

True/False 2. The following statements print a = 0:

```
int a = 9;
a-=--a;
printf("a = %d\n",a);
```

True/False 3. The following statement will print "hello!":

```
if ('a'<'A' && 'b'<'B' || '1'>'2') printf("hello!");
```

True/*False* 4. If you want to create a file in UNIX called myfiles.txt from the results of an ls command you may use the following: `ls < myfiles.txt` (assume that myfiles.txt does not exist)

True/*False* 5. The value of x after the following calculation is 5.500000:

```
Float a=33, b=6, x;
x = (int) a/(int) b;
```

True/*False* 6. The correct way to define the symbol "TRUE" as the value 1 is

```
#define TRUE = 1;
```

True/*False* 7. If A is declared as `int A[5] = {1, 7, 11, 21, 35}` then the value of A[5] would be 35.

True/False 8. Suppose *function1()* is called in *main()* as shown below. If this is an acceptable call to *function1()*, then the only parameter of *function1()* is an array of characters:

```
main()
{
     char x[10][20];
     function1(x[5]);
}
```

True/*False* 9. The following code loops exactly 1 time:

```
#define TERMINATION 6

int x = 5;
while (x<=TERMINATION)
{
      printf(“%d\n”,x);
      x++;
}
```

True/False 10. The following code loops exactly 1 time:

```
#define TERMINATION 6

int x = 6;
do{
      printf(“%d\n”,x);
      x++;
}while (x<TERMINATION);
```

Problem 2 (21 points)

Unix Commands (3 pts. each)

(a) The following command copies the *oldfile* into the *newfile* (True/*False*).

```
cp newfile oldfile
```

(b)
```
-rw-------  1 bhumana ece  6135 Feb 19 1999 f
drwxr-xr-x  2 bhumana ece  512 Mar 25 1999 hw3exit
drwx------  2 bhumana  ece  512 Dec 13 16:32 mail
drwxr-xr-x  6 bhumana ece  1024 Sep 18 15:31 math577
-rw-------  1 bhumana ece  6135 Feb 19 1999 n
```

How would you change the access permissions for *hw3exit* to have the same permissions as the file *mail*?

chmod g-xr
chmod o-xr

(c) What is the command to create a sub directory named ece275 in your current working directory?

mkdir ece275

(d) Create a pipe using UNIX commands so that you can view every line of code containing the variable `chart` in the file *tictactoe.c*.

cat tictactoe.c|grep choice

(e) What is the difference between 1 and 2?

1. command
2. command &

#1 is executed in the foreground while #2 is executed in the background.

(f) If the command `ls|wc>newfile` is executed, what will it do? Assume that the file *newfile* does not exist.

The command combination takes a list of the current directory and counts the number of words in that list, in essence counting the non-hidden directories and files within the current directory, and then puts that number into a newly created file, newfile.

(g) After typing `ps` at the command prompt, the following is displayed:

```
  PID TT    STAT    TIME     COMMAND
56741  p0  Ss      0:00.12 -tcsh (tcsh)
56802  p0  S       0:00.00 sleep 100
56803  p0  R+      0:00.00 ps
```

What is the command to terminated `sleep 100` before it finishes execution?

kill 56802

Problem 3 (39 pts.)

Arrays and Pointers

Refer to the program `trunc.c` below for the questions that follow:

```
1     #include <stdio.h>
2     #define SIZE 5
3     float array1[SIZE] = {1.75, 2.33, 3.1, 4.007, 5.76};
4
5     float f_truncate(float x);
6
7     void main()
6     {
8     int count1 = 0;
9     while (count1 < SIZE)
10          {
11               printf("%.2f %.2f\n", f_truncate(array1[count1]),
                 array1[count1]);
12               count1++;
13          }
14    }
15
16    float f_truncate(float x)
17    {
18          x=(int) x;
19          return (x);
20    }
```

(7 pts.)
(a) If you compile and execute `trunc.c`, what is displayed on the screen?

1.00 1.75
2.00 2.33
3.00 3.10
4.00 4.00
5.00 5.76

(6 pts.)
(b) You have decided to use a for-loop instead of a while-loop in lines 9 through 14. The first thing you do is delete line 12. Re-write line 9 with a `for()` statement using the variable count1:
for (count1 = 0 ; count1 < SIZE ; count1++)

(8 pts.)
(c) Re-write two lines of `trunc.c` so that it will correctly display the truncated values of the elements of the array `{1.76, 4.64, 7.45}`. Indicate the line number of each line that you re-write followed by the revision itself:

2 #define SIZE 3
3 float array1[SIZE] = {1.76, 4.64, 7.45};

(12 pts.)
(d) You've decided to change line 11 to
`printf("%.2f  %.2f",f_truncate(&array1[count1]),array1[count1]);`

Re-write the function `f_truncate()`, in lines 16 through 20, to accept type pointer-to-float as its argument. Notice: this will result in altering parts of lines 16, 18 and 19, but no additional lines will be added.

*16 float f_truncate(float *x)*
17 {
*18 *x=(int) *x*
*19 return (*x);*
20 }

(6 pts.)
(e) If you compile and execute trunc.c after the revisions made in parts (c) and (d), what is displayed on the screen?

1.00 1.00
4.00 4.00
7.00 7.00

Problem 4 (20 pts.)

Multiple Choice
(4 pts. each)

1. What does the following program segment do?

```
int a[10], i;
for (i = 1 ;  i<11 ; i++)
        {
        a[i-1]=i;
        i++;
        }
```

a. Creates the array 'a' where every even element has an odd value
b. Creates the array 'a' where every element except the first equals 2
c. Creates the array 'a' where every element equals twice its index
d. Creates the array 'a' where every odd element has an even value
e. None of the above

2. What would the output of the following C function include?

```
void twilightZone(void);
{
        int i, j, n = 5, count = 0;
        for (i = 5; i>0; i--)
        {
                for (j = i; j>0; j--)
                {
                        count++;
                        printf("count = %d\n", count);
                }
        }
}
```

a. The first line that will print is count = 5
b. The first line that will print is count = 0
c. The last line that will print is count = 1
d. The function will not compile because twilightZone is not a valid name for C functions
e. A constant value 5 is printed because each time j is decremented, count is incremented

3. What will be the output of the following program?

```
main()
```

```
{
        int a = 10;
        int b = 20;
        swapValue(a,b);
        printf("a = %d", a);
        printf("b = %d", b);
}

swapValue(int x, int y)
{
        int t;
        t = x;
        x = y;
        y = t;
        printf(" x = %d", x);
        printf(" y = %d", y);
}
```

a. x = 10 y =20 a = 10 b = 20
b. a =10 b = 20 x = 20 y = 10
c. x = 20 y = 10 a =10 b = 20
d. x = 20 y = 10 a = 20 b = 10
e. Does not print a and b because swapValue doesn't return any values

4. What will be the output of the following program?

```
main()
{
        int arr[]={0,1,2,3,4};
        int i;
        for (i = 0; i<5; i ++)
        {
                printf("%d", arr[i]);
                i ++;
        }
}
```

a. 0 2 4
b. Gives a compilation error
c. 0 3
d. 0 1 2 3 4
e. 0 1 3

5. If a,b and c are declared by the following statements:
int a = 5, b =2;
float c;
what will result after execution of the following statement?

c = 2 * (float) a++ / (float) b--;

a. a = 5, b = 2, c = 5
b. a = 6, b = 1, c = 12
c. a = 5, b = 2, c = 12
d. a = 6, b =1, c = 5
e. a = 5, b = 2, c = 2.5

Problem 3 (36 pts.)

Pointers

(30 pts.)
(a) Fill in the blanks with the appropriate answers.

```
#include <stdio.h>

void order( double _______, double _______ )
{
/*Function to order two numbers at a time*/
double temp; /*temporary variable to hold data during swap*/

/*Compares the numbers pointed to by smp and lgp and switches if
necessary*/
if (*smp>*lgp) {
      temp = *smp;
      *smp = *lgp;
      *lgp = temp;
}
}

void order3( double *num1, double *num2, double *num3)
{
/*Function to order three numbers in the ascending order*/

/*Calls the function “order” to order two numbers at a time*/
order( _________ , __________ );
order( _________ , __________ );
order( _________ , __________ );
}

int main(void)
{
/*Program to sort three numbers in the ascending order*/

double num1, num2, num3; /*numbers to be ordered*/

printf(“Enter three numbers separated by blanks>”);
scanf(“%lf %lf %lf”, &num1, &num2, &num3);

/*Calls the function “order3” to order the three numbers*/
order3( _________ , _________ , __________ );

printf(“The numbers in ascending order are %.2f, %.2f,
%.2f\n”,num1, num2, num3);
return 0;
}
```

(6 pts.)
(b) The following function displays a truncated value of a floating-point number:

```
void f_truncate(float x)
{
x = (int) x;
printf("%.2f\n",x);
}
```

In the following `main()` function, `f_truncate()` is called to display the truncated values of the *numbers* array. However, after this display, the *numbers* array still has its original values.

```
1       main()
2       {
3       float numbers = {1.7, 2.8, 5.63};
4       int count = 0;
5
6       for (count=0; count<3; count++)
7       truncate(numbers[count]);
8       }
```

Re-write the 5 lines of `truncate()` so that line 7 can be changed to `truncate(&numbers[count]);` and the truncated values are retained in *numbers* when the code is executed: