You may take this test with you afterwards, but you must turn in your answer sheet.

This test has the following sections:

I. True/False .................. 64 points; (32 questions, 2 points each)
II. Multiple Choice............. 36 points; (6 questions, 6 points each)

---------------
100 points total

This test is worth 10% of your final grade. You must put your answers on the bubble form. This test is open book and open notes, but no computers. For the multiple choice problems, select the best answer for each one and select the appropriate letter on your answer sheet. Be careful - more than one answer may seem to be correct. Some questions are tricky.

**True/False: (2 points each)** On your bubble form fill out A for true and B for false.

T F 1. The following UNIX command will compile the stuff.c program and create an executable called stuff:

   gcc -o stuff.c stuff.c

T F 2. The following UNIX command gives a directory listing:

   dir

T F 3. If a function returns an int, then the return value must be stored into an int variable and cannot simply be ignored.

T F 4. A do loop is the best type of loop to use to display a menu and prompt for user input.

T F 5. Any code that tests multiple values of a single variable using multiple if-else-if statements in C can be written using a switch-case statement instead.

T F 6. Any code written in C using one of the three looping structures (while, do, for) can be equivalently rewritten using any of the other looping structures.

T F 7. Any single if-else statement can always be rewritten using two if statements.

T F 8. Indentation helps us as humans understand programs in C, but does not in any way change the meaning of a program as far as the compiler is concerned.

T F 9. Reference parameters in C must be passed with an asterisk.

T F 10. Reference parameters in C must be caught with an ampersand and used with an asterisk.

T F 11. Reference parameters in C++ must be caught with an ampersand.

T F 12. Two different functions in C can have the same name, as long as the types or number of parameters are different. **Not in C, though this is true in C++**

T F 13. Two different functions in C++ can have the same name, as long as the return type of the functions are different.
14. Parameters in C++ can have default values.
15. Consider the method used in class to count the total number of students, where in each "round" pairs of students combine their numbers and one of the students in each pair sits down. To count 40 students we would need 6 rounds.
16. A function with a return type of void can still have a return statement in it.
17. The system("pause"); command works on windows but not in UNIX.
18. Consider multiple if-else-if statements used to assign a letter grade ('A'..'F') based on a numerical score (0..100). The successive cases in this code should be indented.
19. Consider multiple if-else-if statements used to handle menu options. The successive cases in this code should be indented.
20. Binary number 101101 in decimal is 43.
21. Decimal value 60 in binary is 111100.
22. Miller's Magic Number is the principle that different people have different memory abilities, however 75% of the population can remember a maximum of 10 unrelated items.
23. A scanf statement that reads in a character can skip leading space simply by including a leading space inside the parenthesis of the format specifier.
24. The following two statements are equivalent:
   ```cpp
cin >> age;
scanf("%d", age);
```
25. Given the declaration
   ```cpp
   char letter;
   ```
   the input statement in C++
   ```cpp
cin >> letter;  // skips leading white space
   ```
   would handle input in the same way as the following input statement in C:
   ```cpp
   scanf("%c", &letter);  // doesn't skip leading white space
   ```
26. The following code will display the text: It is: A
   ```cpp
   char letters[] = {"Able was I ere I saw Elba"};
   printf("It is: %c", letters[1]);
   ```
27. Assume the code in C shown below, where function swapValues is called.
Output of this segment of code is:  Values are: 3 5

```c
void swapValues(int num1, int num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}

int x = 3;
int y = 5;
swapValues( x, y);
cout << "Values are: " << x << " " << y << endl;
// ... other code
```

28. Assume the code in C++ shown below, where function swapValues is called.
Output of this segment of code is:  Values are: 5 3

```c
void swapValues2(int &num1, int &num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}

int x = 3;
int y = 5;
swapValues2( x, y);
cout << "Values are: " << x << " " << y << endl;
// ... other code
```

29. Assume the code in C shown below, where function swapValues is called.
Output of this segment of code is:  Values are: 3 5

```c
void swapValues(int &num1, int &num2)
{
    int temp = *num1;
    *num1 = *num2;
    *num2 = temp;
}

int x = 3;
int y = 5;
swapValues( *x, *y);
cout << "Values are: " << x << " " << y << endl;
// ... other code
```
30. The section of C++ code shown below would compile and run and give as output:

```cpp
bool z = false;
if (z = true)
    printf("Yes ");
else
    printf("No ");
printf("Done \n");
```

31. The following code segment gives the following output:

```cpp
char c='H';
switch (c){
    case 'H': printf("H");
    case 'e': printf("H");
    case 'y': printf("H");
    break;
}
printf("ey");
```

32. The output of the following code is:

```cpp
int x = 4;
if (x > 5)
    printf("It is ");
    if (x < 3) {
        printf("less than 3 ");
    }
else
    printf("Larger ");
printf("Done ");
```
Multiple Choice (6 points each)

33. Consider function $looping1$ shown below. For positive numbers, how would you best describe its return value?

\begin{verbatim}
int looping1(int a, int b)
{
    int answer = 0;
    int x = 0;
    while( true ) {
        if( x < b ) {
            answer += a;
            x++;
        } else {
            break;
        }
    }
    return answer;
}
\end{verbatim}

(a) $a + b$
(b) $a \times a$
(c) $a \times b$
(d) $a$
(e) None of the above

34. Consider function $looping2$ shown below. Assume that sometimes it is called with a single parameter, as in:

\begin{verbatim}
looping2( 1)
\end{verbatim}

and assume that when there are two parameters the second parameter is always a 1, as in:

\begin{verbatim}
looping2( 2, 1)
\end{verbatim}

What does it calculate?

\begin{verbatim}
double looping2(int y, double x=3.14159)
{
    return x * y * y;
}
\end{verbatim}

(a) When there is only one parameter it is the area of a square where the first parameter is the size of the square, and when there are two parameters the area of a circle where the first parameter is the radius.
(b) When there is only one parameter it is the area of a circle where the first parameter is the radius, and when there are two parameters it is the area of a square where the first parameter is the size.
(c) When there is only one parameter it is the area of a circle where the first parameter is the radius, and when there are two parameters it is the area of a rectangle where the two parameters are the height and width
(d) It is always the area of the circle where the first parameter is the radius.
(e) None of the above
35. What is the output from the following C++ code segment?

```cpp
class values[] = "aACCeE";
int sum = 0;
for (int i=0; i<6; i+=2) {
    sum = sum + values[ i] - values[ i+1];
}
sum = sum / 3;
cout << sum << endl;
```

a) 32  
b) 64  
c) 67  
d) 166  
e) None of the above

36. Consider the code shown below. If its output is:

```
12  16  20
15  20  25
```

what are the values for variables start, end, first and last?

a) start = 3, end=4, first=5, last=6  
b) start = 6, end=5, first=4, last=3  
c) start = 4, end=3, first=6, last=5  
d) start = 5, end=6, first=3, last=4  
e) None of the above

37. Consider the code shown below. Assuming that parameter size contains the size of the array, what ends up in array letters?

```cpp
void changeUp( char letters[], int size) 
{
    for ( int i=0; i<size; i++) {
        char temp = letters[i];
        letters[ i] = letters[ size - i - 1];
        letters[ size - i - 1] = temp;
    }
}
```

a) the letters from parameter letters in reverse order  
b) the letters from parameter letters in their original order  
c) the letters from parameter letters in reverse order when the length of letters is odd  
d) the letters from parameter letters in reverse order when the length of letters is even  
e) None of the above
38. What is the output of the following C++ program?

```cpp
#include <iostream>
using namespace std;

int x=3, y=7;

void confuse1(int y, int &x)
{
    x++;  // x = 4
    y++;  // y = 8
}

void confuse2(int *a, int b)
{
    x = b++;  // x = 6, b = 4
    y = ++(*a);  // y = 5
}

void confuse3(int &a, int *b)
{
    a = (*b)++;  // a = 7, b = 6
}

int main()
{
    int y=5;
    confuse1( x, y);  // x = 5, y = 8
    confuse2( &x, y);  // x = 6, y = 7
    confuse3( x, &y);  // x = 7, y = 7
    printf("x + y = %d \n", x+y);
    return 0;
}
```

a) x + y = 7
b) x + y = 10
c) x + y = 13
d) x + y = 14
e) None of the above