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. 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:
   
g++ -o stuff stuff.c

T F 2. A while loop is the best type of loop to use to display a menu and prompt for user input.

T F 3. Any program using nested if-else statements can always be rewritten using only non-nested if statements.

T F 4. Programs are broken up into functions because it makes programs run faster.

T F 5. A C program that runs within a certain Integrated Development Environment (IDE) will always run as-is in any other IDE that also includes a C compiler.

T F 6. Any library in C can also be used in a C++ program, but the #include statement is just a little bit different.

T F 7. A lower-case character stored in variable c can be converted to upper case using the following code:
   
c = c + (\text{'A'} - \text{'a'});

T F 8. A single line of code can give multiple lines of output.

T F 9. Multiple lines of code can give a single line of output.

T F 10. Variable names can contain a space as long as the words on both sides of the space are not reserved words.

T F 11. A function that has exactly two parameters of type float can be called and sent two int parameters.

T F 12. A typecast can be used to convert an int into a char.
13. The following runs and compiles in C:

\[
\begin{align*}
\text{int } x &= 3, y = 5, z = 7; \\
x &= y+++z;
\end{align*}
\]

Some of the following questions ask whether something is valid. A segment of code is valid if it would compile and run within the larger context of a program that itself has no errors.

14. The following array declaration is valid:
\[
\text{int name[]} = \{12345\};
\]

15. The following array declaration is valid:
\[
\text{char name[]} = \{1, 2, 3, 4, 5\};
\]

16. The following array declaration is valid:
\[
\text{char name[]} = \text{"12345"};
\]

17. A compiler will reserve 5 characters for the following array:
\[
\text{char name[]} = \text{"12345"};
\]

18. The following two statements are equivalent:
\[
\begin{align*}
\text{cin } &\triangleright\triangleright \text{ age;} \\
\text{scanf("%d", age);}
\end{align*}
\]

19. Given the declaration
\[
\text{char letter;}
\]
the input statement in C++
\[
\text{cin } \triangleright\triangleright \text{ letter;}
\]
would handle input in the same way as the following input statement in C:
\[
\text{scanf("%c", &letter);} \\
\]

20. The following code will display the text: It is: A
\[
\begin{align*}
\text{char letters[]} = \{\text{"Able was I ere I saw Elba"}\}; \\
\text{printf("It is: %c", letters[1]);}
\end{align*}
\]

array indices start at 0
21. 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
```

22. 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
```

23. Consider code used to display a table of numbers where some are 1 digit numbers, some are 2 digit numbers and some are 3 digit numbers, where we want the numbers to be right-justified and lined up in columns. (An example of this is a 10 x 10 multiplication table.) The best way to do this is to use if statements to test the magnitude of each number to be printed, printing extra spaces for smaller numbers.

To see if variable x is equal to or greater than variable y, we could use:

```c
if( x >= y ) {
    System.out.println( "x => y");
}
```

Also accepted:

```c
if( x > y ) {
    System.out.println( "x => y");
}
```

25. Indentation in a C/C++ program can help the C/C++ compiler figure out what some code means where otherwise it would be confused.

26. If two C++ functions had the return types, names and parameters as shown below it would cause a compiler error. Assume the other parts of the functions exist and are written correctly.

```c
int f1()

int f1( char c)
```
27. If two C++ functions had the return types, names and parameters as shown below, the compiler would always be able to tell them apart. Assume the other parts of the functions exist and are written correctly.

```cpp
int f3( int x)
char f3( int y)
```

28. Any C/C++ program that could be written using functions could be rewritten with all the code in `main()`.

29. The section of code shown below would compile and run and give as output: 2 Done

```cpp
int x = 8;
int y = 3;
printf("%d", x/y);
printf(" Done \n");
```

30. The section of C++ code shown below would compile and run and give as output: Yes Done

```cpp
boolean z = false;
if( z = true)   // Also accepted
    System.out.println("Yes ");
else
    System.out.println("No ");
System.out.println("Done");
```

31. The following code prints the words: Hey there

```cpp
char c='H';
switch (c){
    case 'H': System.out.print("H");
    case 'e': System.out.print("e");
    case 'y': System.out.print("y");
    break;
}
printf(" there");
```

32. The output of the following code is: Larger Done

```cpp
int x = 7;
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 \textit{looping1} shown below. For positive numbers, how would you best describe its return value?

\begin{verbatim}
int looping1( int x, int y)
{
    int answer = 0;
    for( int s = 0; s < x; s++ ) {
        for( int z = 0; z < y; z++ ) {
            answer++;
        }
    }
    return answer;
}
\end{verbatim}

a) $x + y$

b) $x \times x$

c) $x \times y$

d) $x$

(e) None of the above

34. Consider function \textit{looping2} shown below. For positive numbers, how would you best describe its return value?

\begin{verbatim}
int looping2( int x, int y)
{
    int answer = 0;
    while( x + y < x \times y ) {
        answer = answer + x;
        y = y - 1;
        x = x + 1;
    }
    return answer;
}
\end{verbatim}

a) $x + y$

b) $x \times x$

c) $x \times y$

d) $x$

(e) None of the above
35. What is the output of the following C program?

```c
#include <stdio.h>

int x = 5;
int y = 1;

int first(int x)
{
    int temp = y + 1;
y = x;
x = temp;
    return x;
}

//end first

void second(int x, int z)
{
    first(x);
    y = z + 1;
}

//end second

int main()
{
    int y = 2;
    int x = first(y);
    second(y, x);
    printf("Answer is: %d", x + y);
    return 0;
}

//end main()
```

a) Answer is: 4  
b) Answer is: 6  
c) Answer is: 7  
d) Answer is: 8  
e) None of the above
36. What is the output of the following C program?

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

int x=6, y=8;

void confuse1(int y, int &x)
{
    x++;
    y++;
}

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

void confuse3(int &a, int *b)
{
    (*b)++;
    a = x;
}

int main()
{
    int x=1, y=3;
    confuse1(x, y);
    confuse2(&x, y);
    confuse3(x, &y);
    printf("x + y = %d \n", x+y);
    return 0;
}
```

a) x + y = 4
b) x + y = 6
c) x + y = 7
d) x + y = 8
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?

```
void changeUp(char[] letters, int size) {
    for (int i = 0; i < size/2; 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. Consider the code shown below. If its output is:

```
| 4 | 12 | 16 | 20 |
| 5 | 15 | 20 | 25 |
| 6 | 18 | 24 | 30 |
| 7 | 21 | 28 | 35 |
```

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

```
for( int i=start; i<=end; i++) {
    for( int j=first; j<last; j++) {
        System.out.printf("%4d", i*j);
    }
    System.out.println();
}
```

a) start = 7, end=3, first=6, last=4  

b) start = 4, end=7, first=3, last=6  

c) start = 7, end=4, first=6, last=3  

d) start = 3, end=7, first=4, last=6  

e) None of the above