You may take this test with you after the test, 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. Every C program can run in a C++ compiler.
T  F   2. Every C++ program can run in a C compiler.
T  F   3. 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   4. A function with a return type of void can still have a return statement in it.
T  F   5. Binary search will still work if data is unsorted, but it takes more comparisons to find the item being searched for.
T  F   6. Selection sort is always at least twice as fast as Bubble sort.
T  F   7. The name of an array is constant address of the first element.
T  F   8. When an array is passed to a function, changes to array elements are not reflected back to the calling part of the code unless the array is declared to be a reference parameter.
T  F   9. Code handling a one dimensional array typically uses a single for loop. Code handling a two-dimensional array typically uses a for loop nested inside a second for loop. This same idea gets extended out to n dimensional arrays.
T  F  10. When a one-dimensional array is passed as a parameter, the type must be given but the number of elements does not have to be given.
T  F  11. & is the address operator in C
T  F  12. * is the dereference operator in C
T  F  13. In C and C++ it is easy to overwrite the end of an array.
T  F  14. All C-strings must be terminated by a NULL character.
T F 15. If two strings are not equal in C, then the `strcmp` function always returns a non-zero positive number.

T F 16. The `strtok` function takes a numerical value stored in a string and converts it to the corresponding number of k bits.

T F 17. In C the `strstr` function can be used to concatenate two strings.

T F 18. The following code segment separates out the first two words from the given text string and stores them into two separate NULL-terminated strings:

```
char text[]="All generalizations are false"; 
char first[81]; 
char second[81];
char *pCurrent = text; 
char *pSpace = NULL; 
int length = 0; 

pSpace = strchr(pCurrent,' '); 
length = pSpace - pCurrent; 
strncpy(first,pCurrent,length); 
pCurrent = pSpace + 1; 
pSpace = strchr(pCurrent,' '); 
length = pSpace - pCurrent; 
strncpy(second,pCurrent,length); 
``` 

T F 19. When we use ++ to increment a pointer variable in C, under some conditions we add 1 to it, but other times we add 2 or 4.

T F 20. If we want to use a single array to store names as letters and ages as integers for a group of people, we must define and use a `struct`.

T F 21. Assume we had the following declarations:

```
struct Person {
  char name[25];
  int age;
};

Person p1 = "Erin", 23; 
Person *pPerson = &p1;
``` 

Would the following code segment be correct?

```
printf("Person is: %s %d", &pPerson.name, pPerson->age); 
``` 

T F 22. Assume that an int array is declared inside of main and passed to function `displayValues`. Within the `displayValues` function declaration the array could be caught as either of the following two:

```
int *pArray; 
int theArray[];
```
T F 23. The following function could be used to add a single integer to a dynamically growing array of integers:

```cpp
void add( int newNumber, int * &pArray, int &size) {
    int *pNewArray = new( int[ size + 1]);
    for( int i=0; i<size; i++) {
        pNewArray[ i] = pArray[ i];
    }
    pNewArray[ size] = newNumber;
    delete( pArray);
    pArray = pNewArray;
    size++;
}
```

T F 24. A struct can be nested inside another struct.

T F 25. Recursion is when a function directly or indirectly calls itself.

T F 26. Output of the code segment shown below is: B

```cpp
char values[]="ABCD";
char *pChar = values;
printf("%c", pChar + 1);
```

T F 27. Output of the code segment shown below is: B

```cpp
char values[]="ABCD";
char *pChar = values;
printf("%c", *pChar + 1);
```

T F 28. Output of the code segment shown below is: B

```cpp
char values[]="ABCD";
char *pChar = values;
printf("%c", *(pChar + 1));
```

T F 29. In a C program with an array declared using square brackets [ ], the size of the array can be a variable.

T F 30. In a C++ program with an array declared using `new`, the size of the array can be a variable.

T F 31. Dynamically allocated memory in a C++ program can be freed up for reuse by using the `delete` command.

T F 32. In a C program the `grow` command can be used to add storage to a one-dimensional array.
Multiple Choice (6 points each)

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

```cpp
int looping1( int a, int b)
{
    int answer = 0;
    int x = 0;
    
    while( true) {
        if( x < b) {
            answer += a;
            x++;
        } else {
            break;
        }
    }
    
    return answer;
}
```

a) a + b  
b) a * a  
c) a * b  
d) a  
e) None of the above

34. Consider function `defaults` shown below. Assume that sometimes it is called with a single parameter, as in:
```cpp
defaults( 1)
```
and assume that when there are two parameters the second parameter is always a 1, as in:
```cpp
defaults( 2, 1)
```
What does it calculate?

```cpp
double defaults( int y, double x=3.14159)
{
    return x * y * y;
}
```

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

```c
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

36. Consider the code segment shown below. If after the function call the value of number has changed, what is the most likely cause?

```c
int number = 5;
printf(" %d", number);
char letters[]="ABCD";
changeLetters( letters);
printf(" %d", number);
```

a) number is a global variable
b) Although number is not passed to function changeLetters, function changeLetters itself calls a second function which changes number
c) Function changeLetters overwrites the end of array letters
d) There is some ASCII control characters that are present in the code even though they are not visible
e) None of the above
37. What is the output of the following C++ program?

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

int x=2, y=6;

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

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

void confuse3(int &a, int *b)
{
    (*b)++;
    confuse2( b, a);
    a = *b;
}

int main()
{
    int x=0;

    confuse1( x, y);
    confuse2( &x, y);
    confuse3( x, &y);
    printf("%d", x+y);
    return 0;
}
```

a) 4  
b) 8  
c) 16  
d) 32  
e) None of the above
38. Consider the function shown at right below. For positive numbers, how would you best describe the return value of calling function first?

\[ \begin{align*}
\text{a)} & \quad x + y \\
\text{b)} & \quad x \times x \\
\text{c)} & \quad x \times y \\
\text{d)} & \quad x \\
\text{e)} & \quad \text{None of the above}
\end{align*} \]

\[
\text{double first( int x, int y)} \\
\quad \{ \\
\quad \quad \text{if( y==0)} \\
\quad \quad \quad \text{return 1.0;}} \\
\quad \quad \text{else if( n>0)} \\
\quad \quad \quad \text{return first( x, n-1) \times x;}}
\]

39. What is the output of the following C program?

```c
#include <stdio.h>

int process( int *pNumber)
{
    int size = *pNumber;
    int offset = 0;
    int total = 0;
    pNumber++;

    while( offset <= size ) {
        total += *(pNumber + offset);
        offset += *(pNumber + offset);
    }

    return total;
}

int main()
{
    int numbers[] = {7,2,4,1,2,6,7};
    int *pNumber = &numbers[0];

    printf(" %d\n", process( pNumber));
    return 0;
}
```

\[ \begin{align*}
\text{a)} & \quad 1 \\
\text{b)} & \quad 5 \\
\text{c)} & \quad 7 \\
\text{d)} & \quad 12 \\
\text{e)} & \quad \text{None of the above}
\end{align*} \]