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 .......................... 24 points; (12 questions, 2 points each)
   II. Multiple Choice................... 75 points; (15 questions, 5 points each)

---------------------------------
   99 points + 1 for free = 100 points total

This test is worth 15% 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. When a problem describes a segment or fragment of code you may assume the rest of the program is correct and would be supplied to make it work.

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

   T  F   1. In the code shown below, the output is:  y is 5
       
       | int x=2, y=4;
       | if ((x >= 3) && (y = 5)) {
       |     printf("y is: %d", y);
       | }

   T  F   2. Two different functions in C can have the same name, as long as the types or number of parameters are different.

   T  F   3. The name of an array is a constant address of the first element.

   T  F   4. & is the dereference operator in C

   T  F   5. & is used in C++ to indicate a parameter is a reference parameter

   T  F   6. * is the address operator in C

   T  F   7. * is used in a variable declaration to indicate the variable is a pointer

   T  F   8. If two strings are not equal in C, then the strcmp function always returns a non-zero positive number.

   T  F   9. When we use ++ to increment a pointer variable in C, sometimes the address it is pointing to gets incremented by 1, but other times it gets incremented by some other number.
For the next three problems consider the following declaration:

```
struct Person {
    char name[25];
    int age;
};
Person p1 = {"Erin", 23};
Person p2 = {"Darin", 37};
Person *pPerson = &p1;
```

T F 10. Would the following code compile and run and allow storing first name and age properly?

```
printf("Enter first name and age: ");
scanf("%s %d", p1.name, *(pPerson.age));
```

T F 11. Would the following code compile and run and allow storing first name and age properly?

```
printf("Enter first name and age: ");
scanf("%s %d", p1.name, &(pPerson->age));
```

T F 12. Would the following code compile and run?

```
p2 = p1;
```

II. Multiple Choice (5 points each)

13. Consider the two program segments shown below:

Option A:

```
void change1( int *x, int y)
{
    *x = *x + 2;
    y = y + 1;
}
int main()
{
    int x=1;
    int y=3;
    change1( &x, y);
    printf("%d", x+y);
}
```

Option B:

```
void change2( int &x, int y)
{
    x = x + 2;
    y = y + 1;
}
int main()
{
    int x=1;
    int y=3;
    change2( x, y);
    printf("%d", x+y);
}
```

Which of the following is the best answer regarding the above two programs A and B?

a) Neither A nor B will compile and run as expected.
b) A will compile and give output of 6, but B will not
c) B will compile and give output of 6, but A will not
d) Both A and B will compile and run and give output of 6
14. Consider the two programs shown below:

Option A:
```c
#include <stdio.h>

int main()
{
    int value;
    printf("Enter value: \n");
    scanf("%d", &value);
    printf("+1 is:%d",value+1);
    return 0;
}
```

Option B:
```c
#include <iostream>
using namespace std;

int main()
{
    int value;
    cout << "Enter value:";
    cin >> value;
    cout << "+1 is:" << value+1;
    return 0;
}
```

Which of the following is the best answer regarding the above two programs A and B?

a) Neither A nor B will compile and run as expected.
b) A will compile and run as expected, but B will not
c) B will compile and run as expected, but A will not
d) Both A and B will compile and run as expected.

For the following two problems, consider the following four alternatives of types of code, where the layout helps convey what the code is doing:

A)
```
if (expression1)
    action1;
else if (expression2)
    action2;
else if (expression3)
    action3;
```

B)
```
if (expression1)
    action1;
if (expression2)
    action2;
if (expression3)
    action3;
```

C)
```
switch (variable) {
    case 1: action1;
    case 2: action2;
    case 3: action3;
    break;
}
```

D)
```
if (expression1)
    action1;
else
    if (expression2)
        action2;
    else
        if (expression3)
            action3;
```

15. Which of the above types of code would be best to use for a program that checks a test score and assigns a letter grade? Select A, B, C or D.

16. Which of the above types of code would be best to use for a program that checks whether input is upper-case, lower-case, or has an even ASCII value? Select A, B, C or D.
17. How many errors would you need to correct in the code shown below in order to get it to compile and run correctly?

```c
int value;

printf("Enter yearly salary: ");
scanf("%c", value);

if( value = 10000) {
    printf("You are making 10000\n");
else
    printf("You are making something else.\n")
}
```

a) 2 or 3 errors
b) 4 or 5 errors
c) 6 or 7 errors
d) 8 or 9 errors

18. What is the output of the code below when called with `f2();`

```c
char * f1(char theWord[], char c) {
    int x = 0;
    int count = 0;
    for( int i=0; i<strlen( theWord); i++) {
        if( theWord[ i] == c) {
            count++;
            if( count > 1) {
                return &theWord[ i];
            }
        }
    }
    return theWord;
}

void f2() {
    printf("%s", f1("dividian", 'i'));
}
```

a) dividian
b) ividian
c) idian
d) ian
Consider the two versions of function add(…) shown below:

**Option A:**
```c
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);
    size++;
    pArray = pNewArray;
}
```

**Option B:**
```c
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);
    (*size)++;
    *pArray = pNewArray;
}
```

19. Given the following code that could be used to call one of the functions above to grow the array:
```c
int numbers[] = {1,3,5};
int size = 3;
add( 7, numbers, size);
```
which of the following are true about the use of code to call Option A and B above?

a) Neither A nor B will compile and run with this code  
b) A will not work, but B will compile and run with this code  
c) A will compile and run, but B will not work  
d) Both A and B will compile and run with this code

20. Consider calling one of the above versions of function add(...) using the following code:
```c
int size = 3;
int *pNumbers = new int( size);  
pNumbers[0]=1;  
pNumbers[1]=3;  
pNumbers[2]=5;  
add( 7, &pNumbers, &size);
```
When using this code, which of the following are true about the code in Option A and B above?

a) Neither A nor B will compile and run with this code  
b) A will not work, but B will compile and run with this code  
c) A will compile and run, but B will not work  
d) Both A and B will compile and run with this code
21. Consider the two algorithms shown below used to create a letter substitution array:

<table>
<thead>
<tr>
<th>Option A:</th>
<th>Option B:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place a random alphabetic character in each row of an array.</td>
<td>Place a <em>unique</em> random alphabetic character in each row of an array.</td>
</tr>
</tbody>
</table>

After creating the substitution array, use the array to substitute the corresponding new character for each character in the original file.

Which of the following is the best answer regarding the above two algorithms A and B if we want to “encode” the file so that it is not readily legible (i.e. you can’t recognize the words), but it can later be decoded after some analysis?

a) Neither A nor B will work  
b) A will work, but B will not  
c) A will not work, but B will  
d) Both A and B will work

22. What is the output from the following code when calling function `pointerIncrementProblem()`?

```c
void foo( int *pArray)
{
    for( int i=0; i<8; i++) {
        *(pArray++) = (*pArray++) + 1;
    }
}

void pointerIncrementProblem()
{
    int size = 8;
    int theArray[8] = {1,3,5,7,9,11,13,15};
    foo( theArray);
    for( int i=0; i<size; i++) {
        printf("%3d", theArray[ i]);
    }
}
```

da) 2 4 6 8 10 12 14 16

23. When a 2-D array is passed to a function, in the function declaration the size of the first parameter may be left blank, but the second dimension must be supplied. Why is this?

a) The total size of the array divided by the size given in the second dimension is used internally to calculate the number for the first dimension, so the user need not supply it.

b) The size of the first dimension is always automatically supplied, even when not specified by the user, since it is included as part of the definition of every array.

c) Code in C can overwrite the end of an array, however for a 2-D array the formula used to find the i-th row needs to know how many columns are on each row.

d) A NULL character is always inserted at the end of every row so that the compiler can tell where one row ends and the next begins, so the size of the first dimension is not necessary.
24. Consider the function shown at right below. For positive numbers, how would you best describe the return value of calling function \texttt{first}?

\begin{verbatim}
int first(int x, int n)
{
    if (n==0)
        return x;
    else if (n>0)
        return first(x, n-1) + 1;
    return 0;
}
\end{verbatim}

a) \texttt{n}  
b) \texttt{x + n}  
c) \texttt{x * x}  
d) \texttt{x^n}

25. Consider the code segment shown below. If after the function call to \texttt{changeLetters(…)} the value of \texttt{number} has changed, what is the most likely cause?

\begin{verbatim}
int number = 5;
printf("%d", number);
char letters[]="ABCD";
changeLetters(letters);
printf("%d", number);
\end{verbatim}

a) \texttt{number} is a global variable instead of a local variable as it should be  
b) Although \texttt{number} is not passed to function \texttt{changeLetters()}, function \texttt{changeLetters()} itself calls a second function which changes \texttt{number}  
c) Function \texttt{changeLetters()} overwrites the end of array \texttt{letters}  
d) There is some ASCII control characters that are present in the code even though they are not visible

26. Consider the struct declaration and code shown at right below intended for setting and displaying employee information for a single employee. Which of the following best describes the correctness of this code segment?

\begin{verbatim}
struct Employee {
    int id;
    float hourlyWage;
    char code;
};

Employee anEmployee;
Employee *pEmployee = &anEmployee;
anEmployee.id = 2547;
anEmployee.hourlyWage = 8.0;
(*pEmployee).code = 'D';

cout << pEmployee.id;
cout << *pEmployee.hourlyWage;
cout << anEmployee->code;
\end{verbatim}

a) The code will compile and run correctly as shown  
b) The code will not compile and has one error  
c) The code will not compile and has two errors  
d) The code will not compile and has three or more errors
27. Consider a maze program segment shown at right. It is similar, though not exactly the same, as what we discussed in class. Note that there are a total of 20 array values of zero. This function will be called as:

```
makeMove( start);
```

What is the output from running this code?

a) The square numbers of the solution path only, in order, displayed once.

b) The square numbers of the solution path in reverse order, displayed once.

c) The square numbers of the solution path in order, along with numbers of extra squares visited along the way.

d) The square numbers of the solution path in reverse order, along with numbers of extra squares visited along the way.

```cpp
int start = 13;
int goal = 56;
int maze[] =
/*       + 0 1 2 3 4 5 6 7 8 9 */
/* 0 */ {1,1,1,1,1,1,1,1,1,1}, /* 10 */ {1,1,1,0,0,0,0,0,0,1}, /* 20 */ {1,1,1,1,0,0,1,1,1,1}, /* 30 */ {1,0,0,0,0,0,0,0,0,1}, /* 40 */ {1,0,1,1,0,1,1,1,1,1}, /* 50 */ {1,1,1,0,0,0,0,0,1,1}, /* 60 */ {1,1,1,1,1,1,1,1,1,1};
int moves[] = {-1,-10,1,10};
int cameFrom[70];
int done = 0;

void makeMove( int current)
{
    cout << current << "", ";
    if ( current == goal) {
        done = 1;
        return;       // found solution,
    }
    for (int i=0; i<4; i++) {
        int next = current + moves[i];
        if ( (maze[next] != 1) &&
        (next != cameFrom[current]) &&
        !done) {
            cameFrom[ next] = current;
            makeMove( next);
        }
    }
} //end makeMove
```