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

This test has 18 multiple choice questions, each worth 5 points, for a total of 90 points.

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.

Multiple Choice (5 points each)

1. Consider the following declaration:

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

Which of the following would compile and run, allowing storing first name and age properly?

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

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

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

d) ```c
printf("Enter first name and age: ");
scanf("%s %d", p1.name, &(pPerson->age ) );
```
2. What is the output from the code segment shown at right below, called with `scope();`?

A) 2
B) 3
C) 4
D) 5

```c
int x=0; // global variable
void s2( int y)
{
    printf("%d", x+y);
}
void s1( int y)
{
    y++;
    int x=y;
    s2( y);
}
void scope()
{
    x=2;
    s1( x);
}
```

3. Consider the two program segments shown below:

Option A:

```c
int c1( int *x, int &y)  
{
    *x = *x + 1;
    y = y + 1;
    return y + *x;
}
int main()
{
    int x=1;
    int y=3;
    printf("%d", c1( &x, y ) );
}
```

Option B:

```c
int c2( int x, int y)
{
    x = x + 2;
    y = y + 1;
    return y + x;
}
int main()
{
    int x=1;
    int y=2;
    printf("%d", c2( 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
4. Consider the two programs shown below:

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

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

Option B:
```
#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.

5. Consider the following statement:
   ```
   grade = (score > 90) ? 'A' : 'B';
   ```
   This is equivalent to which of the following sections of code?

A) 
```
if( score > 90)  
    grade = 'A'- 'B'; 
else 
    grade = 'B'- 'A';
```

B) 
```
if( score < 90)  
    grade = 'A';
else 
    grade = 'B';
```

C) 
```
if( score >= 90)  
    grade = 'A';
else 
    grade = 'B';
```

D) 
```
if( score <= 90)  
    grade = 'B';
else 
    grade = 'A';
```
6. Consider the code segment shown below:

```c
int i=1;
do {
    printf("%d ", i++);
} while( i<=5);
```

Which of the following two code segments will give the same output as the above code?

Option I:

```c
int j=0;
while( j<5) {
    printf("%d ", ++j);
}
```

Option II:

```c
int k;
for( k=1; k<=5; k++) {
    printf("%d ", k);
}
```

a) Neither I nor II will give the same output.
b) I will give the same output, but II will not
c) II will give the same output, but I will not
d) Both I and II will give the same output.

7. Which of the above types of code would be best to use for a tax rate program that takes an income and matches it against a range of possible tax rates? Select A, B, C or D.

8. Which of the above types of code would be best to use for the section of a program than handles menu options, where some of the menu options result in calling the same code? Select A, B, C or D.
9. How many errors would you need to correct in the code shown below in order to get it to compile and run correctly?

```c
char value;

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

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

a) 1 or 2 errors  
b) 3 or 4 errors  
c) 5 or 6 errors  
d) 7 or 8 errors

10. What is the output of the code below when called with:
```c
char aString[] =  "One.Two.Three" ;
char *pString = aString;
f10( pString, '.');
```

```c
void fc( char *pC)
{
    *pC = '\0';
}

void f10( char *aString, char c)
{
    char *pChar1 = strchr( aString, c);
    fc( pChar1++);
    char *pChar2 = strchr( pChar1, c);
    fc( pChar2++);
    printf("%s %s\n", pChar1, pChar2);
}
```

a) One Two Three  
b) One.Two.Three  
c) Two.Three  
d) Two Three

11. 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\textsuperscript{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.
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;
}
```

12. 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 to grow the array
- b) A will not work, but B will compile and run with this code to grow the array
- c) A will compile and run to grow the array, but B will not work
- d) Both A and B will compile and run with this code

13. 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
14. Consider the function shown at right below. For positive numbers, how would you best describe the return value of calling function \textit{first} with parameters \( x \) and \( n \)?

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

a) \( n \)  
b) \( x + n \)  
c) \( x \times x \)  
d) \( x^n \)

15. Consider the code segment shown below. If after the function call to \textit{changeLetters(…)} the value of \textit{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) \textit{number} is a global variable instead of a local variable as it should be  
b) Although \textit{number} is not passed to function \textit{changeLetters()}, function \textit{changeLetters()} itself calls a second function which changes \textit{number}  
c) Function \textit{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
16. Consider the code shown at right, similar to the maze program shown as a video lecture and discussed in class. This function will be called as: 
\[
\text{makeMove( start);}
\]
What is the output from running this code?

\textbf{a)} The square numbers of the solution path only, in order, displayed once.
\textbf{b)} The square numbers of the solution path in \textit{reverse} order, displayed once.
\textbf{c)} The square numbers of the solution path in order, along with numbers of extra squares visited along the way.
\textbf{d)} Numbers printed in an infinite loop

17. Assume the square at position 23 is changed from 0 to 1. Then what is the output from running this code?

\textbf{a)} The square numbers of the solution path only, in order, displayed once.
\textbf{b)} The square numbers of the solution path in reverse order, displayed once.
\textbf{c)} The square numbers of the solution path in order, along with numbers of extra squares visited along the way.
\textbf{d)} Numbers printed in an infinite loop

18. Assume all the squares are set to 0, except the ones around the border. What would be the output if the start is set to 58 and goal is set to 11?

\textbf{a)} The square numbers of the solution path only, in order, displayed once.
\textbf{b)} The square numbers of the solution path in reverse order, displayed once.
\textbf{c)} The square numbers of the solution path in order, along with numbers of extra squares visited along the way.
\textbf{d)} Numbers printed in an infinite loop

```cpp
int start = 11;
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 */   {1,0,0,0,0,0,0,1,1,1,1,}
    /*20 */   {1,1,1,0,1,0,1,1,0,1,1,}
    /*30 */   {1,0,0,0,0,0,0,0,0,1,1,}
    /*40 */   {1,0,1,1,0,1,1,1,1,1,1,}
    /*50 */   {1,1,1,0,0,0,0,1,1,1,1,}
    /*60 */   {1,1,1,1,1,1,1,1,1,1,1,1};
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
```