You may take this test with you afterwards, but you must turn in your answer sheet. This test has 25 multiple-choice questions, worth 4 points each, for a total of 100 points.

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 and some questions are tricky. When a section of code is described as a program segment you can assume it is placed in the context of a program that is otherwise correct and includes all declarations and system libraries needed to make it work.

1. As discussed in class, in a C/C++ program the return type of main() should be:
   A) void
   B) int
   C) float
   D) It doesn’t matter, as long as the declared type is consistent with the actual value being returned.

2. In what sort of situation might a switch-case statement be preferred over multiple if-else statements?
   A) When choosing which hourly wage to assign to a person
   B) When validating whether or not user input of a character is in the range 'A'..'O'
   C) When computing a test grade ('A','B','C','D' or 'F') based on a score 0..100
   D) When selecting among menu options

3. What is the point of the video shown in class of a man eating a bicycle?
   A) To solve a problem try and find solutions to a similar problem
   B) Bigger problems should be broken down into smaller problems
   C) Parts of some problems can’t be solved without a computer
   D) It is helpful to view problems from different perspectives

4. Consider the program or program segment shown at left below. Which of the options (at right below) is the best answer regarding this program or program segment?

   ```c
   #include <stdio.h>
   int main()
   {
     printf("End \n")
     return 0
   }
   ```

   A) It will not compile
   B) It will compile but will crash when it runs
   C) It will compile and run, but will give unexpected results
   D) It will compile and run as expected
5. Consider the program or program segment shown at left below. Which of the options (at right below) is the best answer regarding this program or program segment?

```
#include <stdio.h>
int main()
{
    printf(Done. \n);
    return 0;
}
```

A) It will not compile  
B) It will compile but will crash when it runs  
C) It will compile and run, but will give unexpected results  
D) It will compile and run as expected

6. Consider the program segment shown at left below. Which of the options (at right below) is the best answer regarding this program segment?

```
#include <stdio.h>
int x;
printf("Enter age: \n");
scanf("%d", &x);
printf("Age is: %d", &x);
```

A) It will not compile  
B) It will compile but will crash when it runs  
C) It will compile and run, but will likely give unexpected results  
D) It will compile and run as expected

7. Consider the program segment shown at left below. Which of the options (at right below) is the best answer regarding this program segment?

```
char c;
printf("Enter letter: \n");
scanf("%c", c);
printf("Letter was: %c", c);
```

A) It will not compile  
B) It will compile but will crash when it runs  
C) It will compile and run, but will likely give unexpected results  
D) It will compile and run as expected

8. Consider the program or program segment shown at left below. Which of the options (at right below) is the best answer regarding this program or program segment?

```
int value;
printf("Enter a number: ");
scanf("%d", &value);
if( value%1 = 0) {
    printf("%d is even", value);
}
else {
    printf("%d is odd", value);
}
```

A) It will not compile  
B) It will compile but will crash when it runs  
C) It will compile and run, but will likely give unexpected results  
D) It will compile and run as expected
9. Consider the program or program segment shown at left below. Which of the options (at right below) is the best answer regarding this program or program segment?

```c
char letter = 'B';
char values[3] = {'C','D','E'};
for( int i=0; i<13; i++ ) {
    values[ i ] = 'A';
}
printf("letter is %c\n", letter);
```

A) It will not compile
B) It will compile but will crash when it runs
C) It will compile and run, but will likely give unexpected results
D) It will compile and run as expected

10. Consider the following statements about using functions in a program:

   I. Functions allows you to reuse code
   II. Functions help you break a problem into smaller pieces
   III. Functions make your program easier to understand

Which of the above are true statements about functions?

A) II only
B) I and II
C) I and III
D) I, II and III

11. Consider the code segment shown below:

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

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

Option I:

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

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.
12. For which of the following cases is a **do-while** loop the most appropriate?
   A) Running code in an endless loop
   B) Displaying a menu and getting the input, verifying it is a valid menu option
   C) Checking a logical condition to decide whether some code should be executed at all
   D) Repeating code a known number of times

13. For which of the following cases is a **for** loop the most appropriate?
   A) Running code in an endless loop
   B) Displaying a menu and getting the input, verifying it is a valid menu option
   C) Checking a logical condition to decide whether some code should be executed at all
   D) Repeating code a known number of times

14. What is the output from the following code?

   ```c
   char c1='D';
   char c2='A';
   printf("%c", (c1+c2)/2);
   ```

   A) There is no output because of a compiler error.
   B) 66
   C) 67
   D) B

15. Consider the program shown at right below used to find the average of input numbers. Values to be averaged are typed in, followed by -1 to indicate the end of the input values. Which line could be either added somewhere in the program or could be used to replace a current line in the program to make this program work correctly?

   ```c
   int sum, count, input;
   scanf("Enter values, then -1 \n");
   while( input != -1) {
     scanf("%d", &input);
     sum += input;
     count++;
   }
   int answer = sum / count;
   printf("Average is %d", answer);
   ```

   ```c
   int answer=1.0*(sum/count);
   sum = sum + input;
   if( input == -1) break;
   count = 1;
   ```

   A) `int answer=1.0*(sum/count);`
   B) `int sum, count, input;`  
   C) `sum = sum + input;`
   D) `if( input == -1) break;`
16. Consider the following statement:
\[
\text{grade} = (\text{score} \geq 90) \ ? \ 'A' : 'B';
\]

Consider the following sections of code, to see if they have the same effect as the above statement.

A)
\[
\begin{align*}
&\text{if} (\text{score} \geq 90) \\
&\quad \text{grade} = 'A'; \\
&\text{else} \\
&\quad \text{grade} = 'B';
\end{align*}
\]

B)
\[
\text{switch (score)} \{
\quad \text{case 100:} \\
\quad \text{case 99:} \\
\quad \text{case 98:} \\
\quad \text{case 97:} \\
\quad \text{case 96:} \\
\quad \text{case 95:} \\
\quad \text{case 94:} \\
\quad \text{case 93:} \\
\quad \text{case 92:} \\
\quad \text{case 91:} \\
\quad \text{case 90: grade} = 'A'; \\
\quad \text{break;}
\quad \text{default: grade} = 'B'; \\
\quad \text{break;}
\}
\]

C)
\[
\begin{align*}
&\text{if} (\text{score} \geq 90) \\
&\quad \text{grade} = 'A'; \\
&\text{if} (\text{score} < 90) \\
&\quad \text{grade} = 'B';
\end{align*}
\]

D)
\[
\begin{align*}
&\text{if} (\text{score} < 90) \\
&\quad \text{grade} = 'B'; \\
&\text{else} \\
&\quad \text{grade} = 'A';
\end{align*}
\]

How many of the above four sections of code (options A,B,C,D) could be used in place of the original statement?

A) one  
B) two  
C) three  
D) four

17. Consider the two design options shown below, to be used in creating a tic-tac-toe program:

I.
\[
\begin{align*}
&\text{displayBoard} \\
&\text{while( ! done) } \\
&\quad \text{promptForMove} \\
&\quad \text{makeMove} \\
&\quad \text{displayBoard}
\end{align*}
\]

II.
\[
\begin{align*}
&\text{while( 1) } \\
&\quad \text{displayBoard} \\
&\quad \text{promptForMove} \\
&\quad \text{makeMove} \\
&\quad \text{if( done) } \\
&\quad \quad \text{break;}
\end{align*}
\]

What is the best description of the above two designs?

A) Only option I is correct  
B) Only option II is correct  
C) Neither one is correct  
D) Both are correct
18. Consider the code below, where one of the four function calls shown at right could be inserted into the underlined section:

```c
void f1(int *p, int *q) {
    *p = *p + 1;
    *q = *q - 1;
}
void f2(int a, int b) {
    a++;
    b--;
}
void f3(int *x, int *y) {
    *x = *x - 1;
    *y = *y + 1;
}
void f4(int *c, int *d) {
    (*c)++;
    --(*d);
}

type of the above four function calls could be used in the underlined space so that when function parameters() is called the program prints out the value 10?
A) One
B) Two
C) Three
D) Four

19. What is the output from the code segment shown at right below, called with scope();

```c
int x = 1; // global variable
void sl(int y) {
    printf("%d", x+y);
}
void s2(int y) {
    x = y++;
    sl(y);
}
void scope() {
    x = 3;
    s2(x);
}```
20. Two functions can have the same name as long as:

A) They have different return types
B) They have one of: different numbers or different types of parameters
C) They have different numbers and different types of parameters
D) They have different return types and one of: different parameter types or different number of parameters

21. What is the output from the program shown at right below?

A) 0
B) 10
C) 45
D) It depends on the input

```c
#include <stdio.h>
int main()
{
    const int Max = 5;
    int values[Max] = {1,3,4,7,9};
    int i=0;
    // Reverse array's elements
    for (i=0; i<Max; ++i) {
        values[i] = values[Max - i];
    }
    // Print numbers
    for (i=0; i<Max; ++i) {
        printf("%d ", values[i]);
    }
    return 0;
}
```

22. What is the output from the program shown at right below?

A) 1 3 4 7 9
B) 9 7 4 3 1
C) 0 9 7 7 9
D) 9 7 4 7 9

```c
#include <stdio.h>
int main(void) {
    const int Max = 5;
    int values[Max] = {1,3,4,7,9};
    int i=0;
    // Reverse array's elements
    for (i=0; i<Max; ++i) {
        values[i] = values[Max - i];
    }
    // Print numbers
    for (i=0; i<Max; ++i) {
        printf("%d ", values[i]);
    }
    return 0;
}
```
23. What is the output from the program shown at right below? The line in bold is what is different from the previous version.

A) 1 3 4 7 9
B) 9 7 4 3 1
C) 0 9 7 7 9
D) 9 7 4 7 9

#include <stdio.h>
int main(void) {
    const int Max = 5;
    int values[Max] = {1,3,4,7,9};
    int i=0;

    // Reverse array's elements
    for (i=0; i<Max; ++i) {
        values[i] = values[Max-i-1];
    }

    // Print numbers
    for (i=0; i<Max; ++i) {
        printf("%d ", values[i]);
    }

    return 0;
}

24. What is the output from the program shown at right below? The line in bold is what is different from the previous version.

A) 1 3 4 7 9
B) 9 7 4 3 1
C) 0 9 7 7 9
D) 9 7 4 7 9

#include <stdio.h>
int main(void) {
    const int Max = 5;
    int values[Max] = {1,3,4,7,9};
    int i=0;

    // Reverse array's elements
    for (i=0; i<Max; ++i) {
        values[i] = values[Max-i-1];
        values[Max-i-1] = values[i];
    }

    // Print numbers
    for (i=0; i<Max; ++i) {
        printf("%d ", values[i]);
    }

    return 0;
}
25. What is the output from the program shown at right below? The lines in bold are what is different from the previous version.

A) 1 3 4 7 9
B) 9 7 4 3 1
C) 0 9 7 7 9
D) 9 7 4 7 9

```c
#include <stdio.h>

int main(void) {
    const int Max = 5;
    int values[Max] = {1,3,4,7,9};
    int i=0;

    // Reverse array's elements
    for (i=0; i<Max/2; ++i) {
        int temp = values[i];
        values[i] = values[Max-i-1];
        values[Max-i-1] = temp;
    }

    // Print numbers
    for (i=0; i<Max; ++i) {
        printf("%d ", values[i]);
    }

    return 0;
}
```