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 ........................................ 28 points; (14 questions, 2 points each)
II. Short Multiple Choice ...... 28 points; (7 questions, 4 points each)
III. Long Multiple Choice..... 40 points; (5 questions, 8 points each)

---------------
100 points total

This test is worth 20% of your final grade. You must put your answers on the answer form using a #2 pencil. You must turn in both the answer form and the test in order to get a grade. Be sure to put your name on your test as well as on your answer form. Be sure to have your UIN written on your answer form under the space for code number, left justified, no spaces. You may assume code is in C++ unless specified otherwise. This test is open book and open notes. You have two hours.

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

T T F 1. If variable number is declared as an int, the following would compile and execute correctly:
cin << number;

T F 2. To check if the value of x is greater than 1 and less than 7 we can use:
   if (1 < x < 7)

T F 3. The break command exits the program currently executing.

T F 4. When writing code to represent a loop that implements counting, a while loop is just as good as a for loop.

T T F 5. A sequence of if statements can always be rewritten as a switch statement.

T F 6. Consider the function declaration shown below:

```cpp
void theFunction( char names[][100], int counter)
{
    // More code in here ...
}
```

Changes to array names will automatically be reflected back to the calling part of the program.

T T F 7. The following code will compile and run:

```cpp
int i = 0;
for (; ; i++) {
    i--;
    cout << i << endl;
}
```
For the following 3 problems, consider the declarations:

```c
struct Employee { 
    int age; 
    float hourlyWage; 
};
Employee anEmployee;
Employee *pEmployee = &anEmployee;
anEmployee.age = 21;
anEmployee.hourlyWage = 8.0;
```

T F 8. The age of anEmployee can be displayed using: `cout << pEmployee.age;`

T F 9. The age of anEmployee can be displayed using: `cout << *pEmployee.age;`

T F 10. The age of anEmployee can be displayed using: `cout << anEmployee->age;`

T F 11. Given the declarations:

```c
char word4[] = {"Elba"};
char *pWord = word4;
```

The statement

```c
cout << pWord[0] << endl;
```

will display the text

E

T F 12. Assume the code shown below, where function `swapValues` is called.

```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
```

Output of this segment of code is: Values are: 3 5
For the next two problems assume the following code is part of a larger program:

```c
// ... other code ...
typedef struct Node * pNode;
struct Node {
    int data;
    Node *pNext;
};
// ... other code here ...

int main()
{
    Node *pHead = NULL;
    int number = 3;
    push( number, pHead);
    // ... other code
}
```

13. The function declaration to push a node on the top of a stack could be:

```c
void push(int value, Node *pHead)
```

14. The function declaration to push a node on the top of a stack could be:

```c
void push(int value, pNode &pHead)
```

II. Multiple Choice: Short Problems (4 points each)

15. Consider this program section that declares and uses a multi-dimensional array of characters:

```c
const int Maximum = 15;
char names[ Maximum][ Maximum];
// ... assume name values are appropriately stored into each array element
cout << names[ 15];
```

What is the output of this section of code?

a) 15 characters, one from the first letter of each name  
b) The last name entered  
c) There is no output since it gives a compiler error  
d) Whatever happens to be in memory after the array  
e) None of the above
16. Consider this program section used to implement a simple calculator for positive integers:

```cpp
switch ( option) {
    case 1: result = operand1 + operand2; // Addition was chosen
    case 2: result = operand1 * operand2; // Multiplication was chosen
    case 3: result = operand1 / operand2; // Division was chosen
    case 4: result = operand1 % operand2; // Modulus was chosen
    default: result = -1; // incorrect input
}
cout << "Answer is: " << result << endl;
```

What will be the result of this section of code?

a) It will not compile  
b) It depends on the values of operand1 and operand2  
c) It will always add operand1 and operand2  
d) It will always give the same answer regardless of the inputs  
e) None of the above

17. What is the output of the program segment below?

```cpp
int i = 2;
if ( i >= 3)
    if ( i < 5)
        cout << "inside if ";
    else
        cout << "inside else ";
    cout << "Done";
```

a) inside if Done  
b) inside else Done  
c) Done  
d) No output due to a compiler error  
e) None of the above

18. Consider the following function:

```cpp
int F( int x)
{
    if ( x < 10)
        return 0;
    else
        return F( x/10)*10 + x%10;
}
```

What is the output if this function is called using:    cout << F(2468);

a) 8  
b) 2  
c) 246  
d) 468  
e) None of the above
19. Consider the following section of code:

```c
int x, y, z;
int *xPtr, *yPtr, **zPtr;
x = 7; y = 5; z = 3;
xPtr = &x;
yPtr = &x;
zPtr = &xPtr;
```

The result of the statement

```
cout << **zPtr;
```

is:

a) 3
b) the address in memory of x
c) the address in memory of xPtr
d) the address in memory of zPtr
e) None of the above

20. Consider the recursive function given below:

```c
void f20( char theWord[])
{
    int theWordLength = strlen( theWord);
    if( theWordLength == 0) {
        return;
    }
    else {
        char c = theWord[ 0];
        strcpy( theWord, theWord+1);
        f20( theWord);
        theWord[ theWordLength-1] = c;
    }
}
```

What ends up in array `theWord` which is passed as a parameter?

a) an empty string
b) the first letter in the original string
c) the last letter in the original string
d) the original string in reversed order
e) None of the above
21. Consider calling the function shown below using:
   \( f21(\text{"Rats live on no evil star"}) \);

   ```c
   void f21( char theString[])
   {
       int x = strlen(theString);
       char c;
       for (int i=0; i<x; i++)
       {
           c = theString[i];
           theString[i] = theString[x-i-1];
           theString[x-i-1] = c;
       }
   }
   ```

   After returning from the function call, the value of `theString` is:

   a) The contents of the original array in reverse order
   b) The contents of the original array in the original order
   c) The contents of the original array with the characters shifted over by one position
   d) The contents of the original array with the characters rearranged so they are no longer recognized as the words in the original phrase.
   e) None of the above

III. Multiple Choice: Short Problems (8 points each)
Choose the best answer for each. *Be careful!*

22. Consider the program given below:

   ```c
   #include <iostream>
   using namespace std;
   int main()
   {
       char words[10][15];
       words[0][0] = '\0';    // NULL
       int row=0;
       cout << "Enter some words: ";
       do {
           cin >> words[ row];
       }while ( strcmp(words[row++], "stop") );
       for (int i=0; i<row; i++)
       {
           cout << words[ i] << " ";
       }
       cout << endl;
       return 0;
   }
   ```

   What is the output when the input is:
   one two three stop

   a) one two three
   b) one two three stop
23. The following is a maze program similar to the ones we discussed in class:

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

int start = 11;
int end = 57;
int maze[] =
/*       + 0 1 2 3 4 5 6 7 8 9 */
/* 0 */   {1,1,1,1,1,1,1,1,1,1},
/*10 */   {1,0,0,0,1,1,1,1,1,1},
/*20 */   {1,1,1,1,1,1,1,1,1,1},
/*30 */   {1,0,0,0,0,0,1,1,1,1},
/*40 */   {1,1,0,1,1,0,1,1,1,1},
/*50 */   {1,1,0,0,0,0,0,1,1,1},
/*60 */   {1,1,1,1,1,1,1,1,1,1};
int moves[] = {-1,-10,1,10};
int cameFrom[70];

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

int main()
{
    makeMove( start);
    return 0;
} //end main
```

What is the output from running this code?

a) The solution path in reverse order
b) Numbers displayed in an infinite loop due to a cycle
c) A bunch of numbers, but not the solution path
d) The solution path with additional other numbers
e) None of the above
Consider the program shown in the two columns below using a singly-linked list to implement a first-in-first-out (FIFO) queue:

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

typedef struct Node * pNode;
struct Node {
    int data;
    Node *pNext;
};

void displayQueue( Node *pHead) {
    if( pHead == NULL) {
        return;
    }
    displayQueue( pHead->pNext);
    cout << pHead->data << " ";
}

void addToTail( int number, pNode &pFront, pNode &pTail) {
    pNode pNewNode = new Node;
    pNewNode->data = number;
    pNewNode->pNext = NULL;
    if( pFront == NULL) {
        pFront = pNewNode;
    }
    else {
        pTail->pNext = pNewNode;
    }
    pTail = pNewNode;
}

int removeFromFront( pNode &pFront, pNode&pTail) {
    pNode pTemp = pFront; int value;
    if( pFront != NULL) {
        value = pTemp->data;
    } else {
        cout << "Invalid. Exiting";
        exit( -1);
    }
    pFront = pFront->pNext;
    if( pFront == NULL) {
        pTail = NULL;
    }
    delete pTemp;
    return value;
}

int main() {
    int number = 0;
    pNode pFront = NULL;
    pNode pTail = NULL;
    int menuChoice;
    do {
        cout << "\n";
        cout << "Select option:\n";
        cout << "1. Add to tail\n";
        cout << "2. Remove from front\n";
        cout << "3. Exit\n";
        cout << "Your choice: ";
        cin >> menuChoice;
        switch( menuChoice) {
            case 1:
                cout << "Enter value: ";
                cin >> number;
                addToTail( number, pFront, pTail);
                break;
            case 2:
                number = removeFromFront( pFront, pTail);
                cout << "Retrieved " << number << endl;
                break;
            case 3:
                cout << "Exiting... ";
                exit( 0);
                break;
            default:
                cout << "Invalid choice. Exiting.\n";
                exit( -1);
                break;
        }
        displayQueue( pFront);
        cout << endl;
    } while ( true);
    return 0;
}
```
Use the code shown above to answer the following two problems:

24. Which function if any in the previously shown program is incorrect?
   
   a) displayQueue
   b) addToTail
   c) removeFromFront
   d) More than one function is incorrect.
   e) None of the above. All the code works correctly.

25. Assuming all the rest of the code is fine, which of the following best describes function displayQueue?
   
   a) It compiles and runs correctly as shown.
   b) It would compile and run correctly if passed pTail, not pHead
   c) It compiles and runs, but it displays queue nodes in the reverse order
   d) It compiles and runs, but does not display the recently added node in the queue.
   e) None of the above.

[See following page for the last problem]
26. *Carefully* consider the C/C++ program given below:

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

struct Node {
    int data;
    Node *pNext;
};

void fcn26a( Node *pHead) {
    if (pHead == NULL)
        return;
    else {
        fcn26a( pHead->pNext);
        cout << pHead->data << " ";
    }
}

Node * fcn26( Node *pHead) {
    Node *pTemp;
    if (pHead->pNext == NULL)  {
        return pHead;
    } else {
        pTemp = fcn26( pHead->pNext);
        pHead->pNext->pNext = pHead;
        return pTemp;
    }
}

int main() {
    int number = 0;
    Node *pHead = NULL;
    Node *pTemp;
    cout << "Enter numbers, then -1: ";
    while ( number != -1) {
        cin >> number;
        if (number != -1) {
            pTemp = new Node;
            pTemp->data = number;
            pTemp->pNext = pHead;
            pHead = pTemp;
        }
    }
    pTemp = pHead;
    pHead = fcn26( pHead);
    fcn26a( pHead);
    return 0;
}
```

If the input is: 1 2 3 4 5 6 7 -1
then the output is:

- a) The reverse of the input, excluding -1
- b) The same as the input, excluding -1
- c) The same as the input, including -1
- d) It does not compile correctly.
- e) None of the above

Infinite loop displaying last two nodes on list, as there is a cycle between the two of them.