Test Instructions

READ THIS NOW!

Failure to read this cover page and follow the described instructions may result in serious repercussions! Failure to read these directions will NOT constitute an excuse or defense.

Fill in your NAME and ID-NUMBER accurately on the test. Failure to do this correctly, will result in an inaccurate score being recorded. Fill in your student information below, sign the pledge and fill in the last 4 digits of your social security number in the box at the bottom of each page.

Answer all questions with respect to material as discussed in class. Note that in any questions/answers which require a distinction between integer and real values, integers will be represented without a decimal point, whereas real values will have a decimal, [1344 (integer), 1344.0 (real)].

All answers on the test must be legible. Do not make assumptions about a problem, if you believe a problem can be interpreted differently ask your instructor. Respond to each question with one answer only, multiple responses and questions with omitted responses will be counted incorrect.

No calculators or electronic devices may be used during this exam. Good Luck!

This is a closed book, closed notes test. It is an honor code violation to discuss (in any form: written, verbal or electronic), any portion of this test with any other students, (regardless of whether they are taking the course or not), until the tests have been returned. It is also an honor code violation to have a copy of this test, (in any form: written, verbal or electronic), in your possession outside of the test examination classroom, without the instructor’s written permission.

Virginia Tech honor code pledge:

“I have neither given nor received unauthorized assistance on this test.”

signature

Programming in C
I. Problem Solving (8 points)

Fill in blanks in the following sentence explaining the programs & algorithms relationship.

All _____ Programs _____ are NOT ___ Algorithms ____ ,
but all ___ Algorithms ___ can be ___ Programs ____ .

II. Input (14 points)

Given the following data file, "MIDTERM.DAT", contents:

```
A B C 123
JGLK
X Y 10.2 65.3 34.7
Q  -456
ZNXW
```

Give a sequence of C/C++ stream input statements so that the following variables get the following values from the above data file (define & use temporary variables, opening the file for input).

<table>
<thead>
<tr>
<th>Variable</th>
<th>char1</th>
<th>gets</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>char2</td>
<td>gets</td>
<td>Y</td>
</tr>
<tr>
<td>Variable</td>
<td>float1</td>
<td>gets</td>
<td>65.3</td>
</tr>
<tr>
<td>Variable</td>
<td>float2</td>
<td>gets</td>
<td>34.7</td>
</tr>
<tr>
<td>Variable</td>
<td>char3</td>
<td>gets</td>
<td>Q</td>
</tr>
<tr>
<td>Variable</td>
<td>int1</td>
<td>gets</td>
<td>-456</td>
</tr>
<tr>
<td>Variable</td>
<td>char4</td>
<td>gets</td>
<td>Z</td>
</tr>
</tbody>
</table>

```c
char tmpch ;
float tmpfloat ;
ifstream midterm ;
midterm.open("MIDTERM.DAT");
if (!midterm)    // check not required
    cout << "* Error MIDTERM.DAT does NOT exist *" << endl;
else {
    midterm.ignore(80, '\n') ; //skip line 1
    midterm >> char1 ;
    midterm.ignore(80, '\n') ; //skip remainder line 2
    midterm >>tmpch>> char2 >>tmpfloat>> float1 >> float2 ;
    midterm.ignore(80, '\n') ; //skip remainder line 3
    midterm >> char3 >>int1 ;
    midterm.ignore(80, '\n') ; //skip remainder line 4
    midterm >> char4 ;
    midterm.ignore(80, '\n') ; //skip remainder line 5
}
```
III. Syntax Diagrams (8 points)

Given the following syntax railroad diagrams:

```
blurb
  ┌───────────┐
  │           │
  │  glurb    │
  │           │
  └───────────┘

glurb
  ┌───────────┐
  │           │
  │ urb       │
  │           │
  │ glop      │
  │           │
  │ urb       │
  │           │
  └───────────┘

glop
  ┌───────────┐
  │           │
  │ X         │
  │           │
  │           │
  │ Y         │
  │           │
  │ Z         │
  │           │
  └───────────┘

urb
  ┌───────────┐
  │           │
  │ 5         │
  │           │
  │ 7         │
  │           │
  │ 9         │
  │           │
  └───────────┘
```

Mark the following “blurb”s as valid or invalid.

<table>
<thead>
<tr>
<th></th>
<th>Valid</th>
<th>Invalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>9Y5</td>
<td>✓ _</td>
<td>______</td>
</tr>
<tr>
<td>X99</td>
<td>_____</td>
<td>✓ _</td>
</tr>
<tr>
<td>7Y5</td>
<td>✓ _</td>
<td>______</td>
</tr>
<tr>
<td>97</td>
<td>_____</td>
<td>✓ _</td>
</tr>
<tr>
<td>5WXY9</td>
<td>_____</td>
<td>✓ _</td>
</tr>
<tr>
<td>Y5X9</td>
<td>_____</td>
<td>✓ _</td>
</tr>
<tr>
<td>7</td>
<td>_____</td>
<td>✓ _</td>
</tr>
<tr>
<td>57Z9</td>
<td>_____</td>
<td>✓ _</td>
</tr>
</tbody>
</table>
IV. C Expressions

(10 points)

Using the following definitions, evaluate the expressions:

const int TRUE = 1;
const int FALSE = 0;
int a = 7, b = 5;
char c = 'B', d = 'A';
int e = FALSE, f = TRUE;

\[
\begin{align*}
\text{a + 28 / b * 4 - 23 / 6} & = 24 \\
6 \times b \% a - b & = -3 \\
(d < c) \&\& !e \| f \&\& (b > a) & = \text{TRUE} \\
a++ \times --b & = 28 \\
(c < d) \| ! (f == \text{FALSE}) & = \text{TRUE}
\end{align*}
\]

V. Selection

(10 points)

What does the following statement(s) do? Circle your answer from the choices below.

if a < b
    if b < a
        cout << 1;
    else
        cout << 2;
else
    if a < b
        cout << 3;
    else
        cout << 4;

A) nothing unless A equals B
B) always prints 4
C) prints 2 if a equals b and 4 otherwise
D) prints 2 if a < b and 1 if b ≤ a
E) prints 2 if a < b and 4 otherwise
F) always prints 2
VI. While Loop  
What will the output be when the following code is executed?  

```c++
int row = 1, column = 3;
while (row <= 3)
{
    column = 3;
    while (column <= 10)
    {
        cout << setw(5) << (row * column) ;
        column = column + 2
    }
    cout << endl ;
    row = row + 1;
}
```

**OUTPUT**

```
3  5  7  9
6 10 14 18
9 15 21 27
```

VII. Programming Errors  
The following program segment, (which tries to determine a person’s optimal weight and output if they are overweight or underweight), has examples of the 3 types of programming errors located within it. Assume the appropriate headers have been included above and all variables have been defined as int. Also assume the user enters the following values when prompted:

```
20  140  5  10
```

Locate, identify and correct each error.

```
cout << “Please enter your age, weight, and height in feet and inches: ” ;
cin >> weight >> age >> feet >> inches ; //Logic error - switch age & weight
diff20 = abs( age - 20 ) ; //possible execution error - division by 0
optimalWt = ((feet * 12) + inches) * 2 / diff20 * abs(age - 30) ; //correct formula
if ( optimalWt > weight ) then //Compilation error - omit then
    cout << “You are ” << optimalWt - weight << “ pounds overweight” << endl ;
else //Logic error - switch over & under
    cout << “You are ” << weight - optimalWt << “ pounds underweight” << endl ;
}
```
VIII. Coding  
(25 points)

Write a C/C++ program that displays all 4 digit numbers in which the product of the digits of the number equals four factorial. Since $4! = 24$ a few examples of 4 digit numbers whose 'digit product' $= 24$ are: 1146, 1226, 2223, ...

The output of the program should look like the following:

All four digit numbers whose digit product $= 4! = 24$:

1146 : $1 \times 1 \times 4 \times 6 = 24$
1226 : $1 \times 2 \times 2 \times 6 = 24$
2223 : $2 \times 2 \times 2 \times 3 = 24$

Note: Leading zeroes do NOT count as digits, i.e. the number 0234 is NOT a 4-digit number but a 3-digit number. Hint: use nested loops.

```c
#include <iostream.h>
#include <stdlib.h>

int main()
{
    int dig1, dig2, dig3, dig4, digProd, num;

    cout << "4 digit nums digit product ==4!==24:" << endl;

    for (dig1=1; dig1<10; dig1++)
        for (dig2=0; dig2<10; dig2++)
            for (dig3=0; dig3<10; dig3++)
                for (dig4=0; dig4<10; dig4++) {
                    digProd = dig1 * dig2 * dig3 * dig4;
                    if (digProd == 24) {
                        num = dig1 * 1000 + dig2 * 100 + dig3 * 10 + dig4;
                        cout << setw(4) << num << "  :" << setw(1)
                             << dig1 << " x " << dig2 << " x " << dig3
                             << " x " << dig4 << " = 24 " ;
                    }
                }

    return (EXIT_SUCCESS) ;
}
```