This test has 6 questions on 9 pages. Be sure your test has them all.

This is an open-book test. You have at least 50 minutes to complete it. That means you should spend no more than 1 minute per point. If the number You may consult any books, notes, or other inanimate objects (other than computers or calculators) available to you. You may use any program text supplied in lectures, assignments, or solutions.

Please write your answers in the spaces provided in the test. Make sure to put your name, login, and lab section in the space provided below. Put your login clearly on each page of this test and on any additional sheets of paper you use for your answers.

Don’t panic. Just read all the questions carefully to begin with, and first try to answer those parts about which you feel most confident. Do not be alarmed if some of the answers are obvious.
PROBLEM 1: (Not all that mysterious)

A. (5 points) Consider the following function:

```cpp
int mystery(int x, int y)
{
    if (x > y)
        return x;
    else
        return mystery(y - x, y);
}
```

For each call below, what value is returned?

I. `mystery(14, 10)`
II. `mystery(-10, -10)`
III. `mystery(2, 8)`
IV. `mystery(12, 42)`

B. (3 points) Consider the following function:

```cpp
void mystery()
{
    char ch;

    cin >> ch;
    if (ch != '\n')
    {
        mystery();
        cout << ch;
    }
}
```

What is printed if the user enters the following input

dog
cat
PROBLEM 2:  (Look out for agents! (5 points))

After executing the following code fragment:

```cpp
tmatrix<int> vals(4,5,1);
for (int i = 1; i < vals.numrows(); i++)
    for (int j = 1; j < vals.numcols(); j++)
        vals[i][j] += vals[i-1][j-1]
```

What is the value of the following variables. If the value is unknown, put “unknown”. If the value may return an error, put “error.”

A. `vals[0,0]`
B. `vals[1,3]`
C. `vals[3,4]`
D. `vals[4,5]`

PROBLEM 3:  (Commentary (3 points))

Suppose that you have just come across the following comment and function prototype in a header file:

```cpp
/*
 * Function: IsOrdered
 * Usage: if (IsOrdered(v) == true) ... 
 * ---------------------------------------------------------------
 * Using a for loop this function compares 
 * each element in the vector to the following 
 * element in that vector; if the two are out 
 * of sequence, the function returns false 
 */
bool IsOrdered(tvector<int> v);
```

What suggestions can you make for improving the comments or the declaration itself?
Some algorithms work with points in 3-dimensional space. A program might represent those points as vectors of length three, so that a variable representing a point might be declared as:

```cpp
tvector<int> pt(3);
```

Alternatively, we might use a struct to define a point as follows:

```cpp
struct Point
{
    Point()
        : x(0), y(0), z(0)
    {}
    Point(double px, double py, double pz)
        : x(px), y(py), z(pz)
    {}
    double x;
    double y;
    double z;
}
```

As a final option, we could use a class to define a point as below:

```cpp
class Point
{
    public:
    Point();
    Point(double px, double py, double pz);

double x(); // returns point’s x value
double y(); // returns point’s y value
double z(); // returns point’s z value

string toString() const;

double distanceFrom(const Point& p) const; // Euclidean distance
void translate(double deltaX, double deltaY, double deltaZ);
    // changes point to (x+deltaX, y+deltaY, z+deltaZ)

    private:
        double myX;
        double myY;
        double myZ;
};
```

Briefly describe an advantage for each of the three representations in C++. 
PROBLEM 5:  *(Head of the Class (10 points))*  
You are given the following struct for storing students' names and grades.

```cpp
struct StudentGrade
{
    StudentGrade() : name("NOBODY"), score(0)
    {
    }
    StudentGrade(string word, double val)
        : name(word), score(val)
    {
    }

    string name;
    double score;
};
```

You also are given a working function that inserts a particular StudentGrade record into a vector in reverse sorted order by score. In other words, the record with the highest score will be first. This function `InsertReverseSorted` is below.

```cpp
void InsertReverseSorted(tvector<StudentGrade>& v, const StudentGrade& s)
{
    int count = v.size();  // # stocks before adding new one
    v.push_back(s);  // vector size is updated
    int loc = count;

    // invariant: loc-1 is index of rightmost unprocessed entry
    // for k in v[loc+1..count], s < v[s]
    while (0 < loc && s.score >= v[loc-1].score)
    {
        v[loc] = v[loc-1];
        loc--;
    }
    v[loc] = s;
}
```

If you are given a file with names and scores in the form

```
<firstname> <lastname> <score>
```

as follows:

```
Smart Student 90
VerySmart Person 93
Super Scientist 97
Joe Average 75
```

Fill in the following function to print out a ranking of the students by score as in:

```
1. Super Scientist 97
2. VerySmart Person 93
3. Smart Student 90
4. Joe Average 75
```
The names are in a file denoted by `filename`. You can and should use the `InsertReverseSorted` function and the `StudentGrade` struct.

```cpp
void PrintRanking(string filename)
{
```
PROBLEM 6: (Not Orange (17 points))

The goal of the Dutch National Flag problem as discussed in class is to take $N$ objects colored red, white, and blue, sort them so that the objects are of the same color are adjacent, with the colors in the order red, white, and blue.

Assume the colors are defined as:

```cpp
enum Color {Red, White, Blue};
```

In class, we solved the two color Reverse Polish Flag (red-white) problem. Below is the code using colors with the loop invariant.

```cpp
void TwoColorFlag(tvector<Color> &flag)
{
    int highRed = 0;
    int lowWhite = flag.size()-1;
    for (int i=0; i < flag.size(); i++)
    // INVARIANT: flag[0...highRed-1]==red - all elements with indices below highRed are red
    // flag[highRed...lowWhite]==unknown - others from highRed to lowWhite are unknown
    // flag[lowWhite+1...size-1] all elements above lowWhite are white
    {
        if (flag[highRed] == Red)
            highRed++;
        else
        {
            Swap(flag, highRed, lowWhite);
            lowWhite--;
        }
    }
}
```

The problem continues on the next page.

Extra credit: Name one member of Duke’s programming team that finished 8th at the ACM International Programming Competition in March. (1 point)

One faculty member in the Computer Science department was a member of a team that finished fourth in the ACM programming contest. Who was it? (1 point) No, it wasn’t me.
A. Write the function Swap used above that takes a vector, two integer indices, and swaps those elements. For example:

Given the following vector \( v \):

<table>
<thead>
<tr>
<th>White</th>
<th>Red</th>
<th>White</th>
<th>Red</th>
<th>Red</th>
</tr>
</thead>
</table>

The call \( \text{Swap}(v, 0, 4) \) should change the vector \( v \) so that it looks like:

<table>
<thead>
<tr>
<th>Red</th>
<th>Red</th>
<th>White</th>
<th>Red</th>
<th>White</th>
</tr>
</thead>
</table>

Write the declaration and implementation of \( \text{Swap} \) below:

B. Now consider the original 3 color Dutch flag problem. Your solution will divide the vector into four sections:

<table>
<thead>
<tr>
<th>RRRRRRRRRR</th>
<th>WWWWWW</th>
<th>???????</th>
<th>BBB BBB</th>
</tr>
</thead>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>highRed</td>
<td>highWhite</td>
<td>lowBlue</td>
</tr>
<tr>
<td></td>
<td>size-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Describe the loop invariant that should be maintained. You should make some reference to the indices highRed, highWhite, and lowBlue. Your invariant should be brief and structured like the invariant in the comments for the 2 color problem.
C. Complete the following function for the three color flag below.

```cpp
void ThreeColorFlag(tvector<Color> &flag)
{
    int highRed = ; // FILL IN VALUE
    int highWhite = ; // FILL IN VALUE
    int lowBlue = ; // FILL IN VALUE

    for (int i=0; i < flag.size(); i++)
    { // FILL IN LOOP AS NECESSARY
        if (flag[highWhite] == Red)
        {
            // CODE
        }
        else if flag[highWhite] == White)
        {
            // CODE
        }
        else
        {
            // CODE
        }
    }
}
```

D. Extra credit: What is the maximum number of swaps that will be done for a vector of size \( n \) for the above algorithm (in terms of \( n \))? (1 point)