NAME: ____________________________________

```
bool areIdentical(const vector<int> & a, const vector<int> & b);
```

Figure 1

1) Write a predicate function that checks whether two vectors are identical (contain exactly the same elements in the same order). A declaration of the function is shown in Figure 1. The function returns true if the two vectors are identical; otherwise it returns false. (25 points)

```
int minValue(vector<int> v);
```

Figure 2

2) Provide an implementation of the minValue function whose declaration is shown in Figure 2 above. The function takes a single argument, which is a vector of *int*. The function returns the smallest *int* that is in the vector. (25 points)

```
bool isLucky(vector<int> & v);
```

Figure 3

3) Implement a function that determines if the vector contains at least one 7 and no 13's. The function returns true if the number 7 appears in the vector one or more times and the number 13 does not appear at all; otherwise it returns false. For example, the function returns false for (6, 4), true for (6, 4, 7) and false for (6, 4, 7, 13). A declaration of the function is shown in Figure 3. (25 points)

4) Write test code that tests every statement in the isLucky function in the previous problem. Express your tests using assertions. (25 points)

5) Write a program that implements Guess-the-Number game. The program should enter a loop that starts by printing “What is the number?” After printing this, it reads the user response. (Use cin >> n to read the user response.) If the user enters a value less than 1776, the program prints "too small" and continues the loop. If the user enters a number larger than 1776, the program prints "too big" and continues the loop. If the user enters the number 1776, the program prints "you got it" and then terminates. (25 points)

```
int countNegatives(int a[ROWS][COLS]);
```

Figure 4

6) Implement a function that counts the number of negative integers in a two-dimensional array with ROWS rows and COLS columns. The function takes a two-dimensional array of *int* and returns the number of negative numbers in it. A declaration of the function is shown in Figure 4. The variables ROWS and COLS are constants defined elsewhere in the program; you don't need to define them, just use them. A declaration of the function is shown in Figure 4. (25 points)

```
int search(const vector<int> & v, int k);
```

Figure 5

7) Implement a function that searches for a given value in a vector of integers. If the value is found, the function returns the index of the value in the vector; otherwise it returns -1. Do not assume the values are in order; do not use binary search. For example, for v = (-2, 4, 18, 6) the function would return 2 for k = 18 and it would return -1 for k = 1. A declaration of the function is shown in Figure 5. (25 points)

```
int binarySearch(const vector<int> & v, int k);
```

Figure 6

8) Implement a function that uses binary search to search for a given value in a vector of integers whose elements are in strictly increasing order. If the value is found, the function returns the index of the value in the vector; otherwise, it returns -1. You can assume that the values passed into the function are in strictly increasing order. For example, for v = (-2, 4, 5, 6) the function returns -1 for k = 2 and 1 for k = 4. A declaration of the function is shown in Figure 6. (25 points)