

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)

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

Figure 2

2) Implement a function that determines if the vector only contains the numbers 7 and 8. The function returns true if each number in the vector is either 7 or 8; otherwise it returns false. A declaration of the function is shown in Figure 2. (25 points)

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

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

Figure 3

4) Provide an implementation of the `maxValue` function whose declaration is shown above. The function takes a single argument, which is a vector of `int`. The function returns the largest `int` that is in the vector. (25 points)

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

Figure 4

5) Write a predicate function that checks whether two vectors are complements in the sense that their elements pairwise add to zero. For example, the vectors $(-2, 3, 0, -7)$ and $(2, -3, 0, 7)$ are complements. If the two vectors have different numbers of elements, then they are not comparable, so return false in this case. A declaration of the function is shown in Figure 1. (25 points)

```
int countOccurrences(int a[ROWS][COLS], int k);
```

Figure 5

6) Implement the function `countOccurrences` whose declaration appears above. The first argument of the function is a 2-dimensional array of integers and the second argument is an integer k . The function returns the number of times k occurs in a . A declaration of the function is shown in Figure 5. The variables `ROWS` and `COLS` are constants defined elsewhere in the program; you don't need to define them, just use them. (25 points)

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

Figure 6

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 returns -1 for $k = 1$ and 2 for $k = 18$. A declaration of the function is shown in Figure 6. (25 points)

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

Figure 7

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 7. (25 points)