10.6 Practice Programs

Sample solutions to these problems will be included by the beginning of 2013 at the following website:

http://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3223/book/

Each of the following problems asks you to write a function. In order to test your function, you must write a main function in addition that calls the function you've written. Make sure to test the function you've written with a comprehensive set of test cases.

1) Write a function that adds one to the variable passed in by reference.

```
void increment(int* ptrValue);
```

2) Imagine a casino where each chip costs 10 dollars. Write a function buyChips that allows a user to buy the maximum amount of chips a certain dollar amount will let them. The function should take in two parameters by reference: the amount of money they have (stored in an integer), and the number of chips they currently have. Your function should change the amount of money to how much is left (less than 10) after the transaction and it should add the appropriate number of chips to the variable storing the number of chips. The prototype for the function is below:

```
void buyChips(int* ptrMoney, int* ptrNumChips);
```

For example, if the variable money stored 97 and the variable chips stored 3, then the function call `buyChips(&money, &chips)` would result in money changing to 7 and chips changing to 12.

3) Write a function that updates the odometer and fuel gauge of a car. The function will take in a reference to the variables storing the odometer and fuel gauge readings, along with a double representing the miles per gallon the car gets and the number of miles the driver intends to go. The function should update the odometer and fuel gauge readings, accordingly. If the number of miles is greater than the user can go with the fuel he has left, then simply have the car drive until empty. The prototype for the function is below:

```
void drive(double* ptrOdom, double* ptrFuel,
           double mpg, double distToDrive);
```

4) Write a function loanUpdate, which takes in a pointer to the current value of a loan, the monthly interest rate on the loan as a decimal and a pointer to the monthly payment on the loan. The function should update both the value of the loan and the monthly payment after that payment has been processed. In particular, the loan value should first accrue interest, then decrease by the payment value. The monthly payment should stay the same, except for the case where the total value of the loan *in a month* is less than the monthly payment. In this case, the monthly payment should update to what the last month's payoff will be. The prototype for the function is below:

```
void makeLoanPay(double* ptrLoan, double monRate,
                 double* ptrMonPay);
```

5) Write a function that takes in an array and an integer, representing the length of the array, and adds 1 to each element in the array. The function prototype is given below.

```
void addOne(int values[], int length);
```

6) Write a function that takes in an array and an integer, representing the length of the array, and reverses the contents of the array. The function prototype is given below.

```
void reverse(int values[], int length);
```

7) Write a function that takes in an array and an integer, representing the length of the array, and shifts each element in the array one slot to the left, and puts the item in index 0 at the end of the array. (This is known as a cyclic left-shift.) The function prototype is given below.

```
void cyclicLeftShift(int values[], int length);
```

8) Write a function that takes in an array, an integer, representing the length of the array, and another integer representing a maximum value and fills the array with random numbers in between 1 and the given maximum value. You may assume that the random number generator has already been seeded. Here is the function prototype:

```
void fillRandomArray(int array[], int length, int maxVal);
```

9) Write a function that takes in an array, and an integer, k, and finds the largest value in the indexes 0, 1, 2, ..., k and swaps that value into index k. For example, if the input array initially stored

index	0	1	2	3	4	5	6	7
array	8	3	9	4	5	1	12	13

and the function call `swapMax(array, 5)`, then the array would appear as follows after the function call:

index	0	1	2	3	4	5	6	7
array	8	3	1	4	5	9	12	13

Here is the function prototype:

```
void swapMax(int array[], int k);
```

10) Using the function from #9 write a function to sort an array into order, from smallest to largest. The function should take in the array and its length as parameters. The prototype is given below:

```
void sort(int array[], int length);
```

Test your function by writing a main that fills an array with random values, prints the array, sorts the array, and then prints it out again.