CS 110 - Programming I

Homework 3

General Instructions

- Every problem needs to be solved in a separate .cpp file
- The name of the cpp file needs to be the number of the problem. For example, the file for the solution to problem 2 should be named 2.cpp
- The first two lines of every one of your solutions should be a comment with your full name and your email address, for example:

```
// Pablo Gallastegui
// pablo.gallastegui@brescia.edu
```

- All the solutions need to be compressed (zipped up) in a single file named as the left half (before the @ symbol) of your email address, for example:

```
pablo.gallastegui.zip
```

- The compressed folder has to be submitted using moodle, before the due date and time.
- After every homework is due, a few students may randomly be selected to explain the way they solved one of the problems.

Recommendations

- You can resubmit the assignment multiple times, so I would recommend submitting early and often. Better to have most of your assignment turned in in case of a technology failure than none at all.
- Due date and time is on a Sunday right before midnight. Although I try to do my best, the chances of getting any questions answered right before the due date and time can be small. Try to solve the homework with plenty of times to ask questions in the forum or, preferably, during class.

- You will always get a better credit for a wrong answer that is yours, than for a right answer that is someone else's. Make sure you add comments explaining your reasoning in code to get partial grade even when the problem does not return the correct result.
- Make sure that your output looks exactly like the output required. While we will have plenty of flexibility during the labs to experiment and explain the route we took, I may not be able to understand the reasoning behind changing the output in a homework, and that could result in lost points.
- Feel free to help each other, but do not copy from each other.
- Test your programs with many different inputs, make sure they return the correct result.

Problems

1. (20 points) Use a one-dimensional array to solve the following problem. A company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9 percent of $5000, or a total of $650. Write a program (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):
    a. $200–299
    b. $300–399
    c. $400–499
    d. $500–599
    e. $600–699
    f. $700–799
    g. $800–899
    h. $900–999
    i. $1000 and over

    The screen dialog should appear as follows:

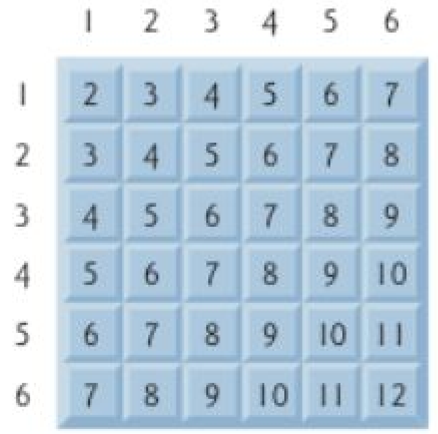
```
Enter employee gross sales (or -1 to quit): 100
Employee commission: $9
Enter employee gross sales (or -1 to quit): 1000
Employee commission: $90
Enter employee gross sales (or -1 to quit): 2000
Employee commission: $180
Enter employee gross sales (or -1 to quit): 4000
Employee commission: $360
Enter employee gross sales (or -1 to quit): -1
Employees in the range:
$200-299: 2
$300-399: 1
$400-499: 0
$500-599: 1
$600-699: 0
$700-799: 0
$800-899: 0
$900-999: 0
```

```
   $1000 and over: 0
```

2. (30 points) Use a one-dimensional array to solve the following problem. Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, validate it and store it in the array only if it isn't a duplicate of a number already read. After reading all the values, display only the unique values that the user entered, sorted in descending order. Provide for the "worst case" in which all 20 numbers are different. Use the smallest possible array to solve this problem. The screen dialog should appear as follows:

```
   Enter 20 integers between 10 and 100
   Number 1: 11
   Number 2: 9
   Invalid number.
   Number 3: 11
   Duplicate number.
   Number 4: 50
   Number 5: 60
   Number 6: 80
   Number 7: 70
   Number 8: 75
   Number 9: 80
   Duplicate number.
   Number 10: 110
   Invalid number.
   Number 11: 11
   Duplicate number.
   Number 12: 20
   Number 13: 30
   Number 14: 35
   Number 15: 100
   Number 16: 99
   Number 17: 98
   Number 18: 72
   Number 19: 73
   Number 20: 15
   The nonduplicate values are: 100, 99, 98, 75, 73, 72, 60,
      50, 35, 30, 20, 15
```

3. (20 points) Write a program that simulates the rolling of two dice. The program should use rand to roll the first die and should use rand again to roll the second die. The sum of the two values should then be calculated. [Note: Each die can show an integer value from 1 to 6, so the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums.] These are the possible combinations of the two dice:



	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Your program should roll the two dice 36,000 times. Use a one-dimensional array to tally the numbers of times each possible sum appears. Print the results. Also, determine if the totals are reasonable, within 5% of the expected value (i.e., there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7). The screen dialog should appear as follows:

```
2: 284 (not reasonable)
3: 600
4: 900
5: 1200
6: 1500
7: 1800
8: 1500
9: 1200
10: 900
11: 600
12: 316 (not reasonable)
```

4. (30 points) Use a two-dimensional array to solve the following problem. A company has four salespeople (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each different type of product sold. Each slip contains the following:
   a. The salesperson number
   b. The product number
   c. The total dollar value of that product sold that day

   Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month is available. Write a program that will read all this information for last month's sales (one salesperson's data at a time) and summarize the total sales by salesperson by product. All totals should be stored in the two-dimensional array sales. After processing all the information for last month, print the results in tabular format with each of the columns represent- ing a particular salesperson and each of the rows representing a particular product. Cross total each row to get the total sales of each product for last month; cross total each column to get the total sales by salesperson for last month. Your tabular printout should include these cross totals to the right of the totaled rows and to the bottom of the totaled columns. The screen dialog should appear as follows:

```
Enter salesperson id (-1 to quit): 1
Enter product number: 1
Enter total dollar value: 900
Enter salesperson id (-1 to quit): 2
Enter product number: 1
Enter total dollar value: 1000
Enter salesperson id (-1 to quit): 1
Enter product number: 2
Enter total dollar value: 100
Enter salesperson id (-1 to quit): 3
Enter product number: 3
Enter total dollar value: 500
Enter salesperson id (-1 to quit): 4
Enter product number: 5
Enter total dollar value: 600
Enter salesperson id (-1 to quit): -1
            Person 1  Person 2  Person 3  Person 4  Totals
Product 1   900       1000      0         0         1900
Product 2   100       0         0         0         100
```

Product 3	0	0	500	0	500
Product 4	0	0	0	0	0
Product 5	0	0	0	600	600
Totals	1000	100	500	600	3200