

Program Set #3

Total Points: 30

Choose 3 of the 6 problems below for full credit. The starred problem must be implemented. See 1436 Grading Guide Sheet for additional grading/submission information. Partial credit will be given. (10 points each)

1. The simplified version of the scoring system in the Olympic Figure Skating competition is as follows:

- Step 1: All 12 judges submit their scores
- Step 2: 9 scores are randomly selected
- Step 3: The highest and lowest score are thrown out from the nine counting judges
- Step 4: The remaining seven scores are averaged

Write a C++ program that reads 12 keyboard scores submitted by 12 judges for the specific event and calculates the final score according to the rules stated above. The program needs to print the results after every step. Each input score is an integer number between 0 and 100 and separated by a space. Error check for valid scores. Output the final score to three decimal places. Create a user defined function. Refer to the sample output below.

Sample Run:

Enter the scores (0-100): 12 49 100 98 99 16 13 100 99 12 99 100

Results:

Step 1: 12 scores: 12 49 100 98 99 16 13 100 99 12 99 100

Step 2: Nine randomly selected scores: 12 100 98 99 16 13 12 99 100

Step 3: Remaining seven scores: 100 98 99 16 13 12 99

Step 4: Final score: 62.429

Note: The program is supposed to run in different ways on the same input in different runs, since random selection at step 2 is being used.

Name the program: ScoreSkatingXX.cpp, where XX are your initials.

2. Write a C++ program that simulates the game show *Deal or No Deal*. For this version of the game, there are 10 possible dollar amounts: \$100, \$500, \$1000, \$5000, \$10000, \$25000, \$50000, \$100000, \$500000, \$1000000 sealed in imaginary briefcases. These dollar amounts are numbered 1 – 10 (i.e. 1 → \$100, 2 → \$500, 3 → \$1 000,..., 10 → \$1000000). Before the game starts, the contestant will have chosen one of the briefcases as his/hers to keep. During the game, some of the ten

possible dollar amounts have been eliminated from the game because the contestant has selected some of the other briefcases and revealed the amounts inside.

At some point, the contestant will stop opening briefcases, and a "Banker" will offer the contestant cash in exchange for what might be contained in his/her chosen briefcase. Then the contestant is asked: "Deal or No Deal?". The program should decide if he/she should choose "Deal" or "No Deal", by calculating the average of the remaining amounts (i.e., all unopened briefcases, including his/her "own" briefcase), and comparing that value to the "Banker's" offer. If the offer is higher than the average, then the player should "Deal" otherwise, the player should say "No Deal".

The user must input a number n (between 1 and 10) which indicates how many cases have been opened so far, followed by a list of n integers between 1 and 10 representing the values in the game that have been eliminated, followed by the "Banker's" offer. For example: 3 2 5 10 300 indicates that briefcases containing \$500, \$10000, and \$1000000 have been eliminated and the Banker's offer is \$300. Assume that no duplicate case numbers are entered for the eliminated values and assume that the "Banker's" offer is an integer greater than 10. The program will print out one of two statements: "Deal!" or "No Deal!". Finally, the program should ask if the user wants to run the program again or quit. Check case. Refer to the sample output below.

Sample Run:

Deal or No Deal?

How many cases have been opened so far: 2

Enter the cases that have been eliminated so far

(1 - \$100, 2 - \$500, 3 - \$1 000,...,10 - \$1000000) on one line: 3 8

What is the Banker's offer: 198000

No Deal!

Run again (Y/N): y

Deal or No Deal?

How many cases have been opened so far: 8

Enter the cases that have been eliminated so far

(1 - \$100, 2 - \$500, 3 - \$1 000,...,10 - \$1000000) on one line: 10 9 8 7 6 5 4 3

What is the Banker's offer: 400

Deal!

Run again (Y/N): N

Name the program: DealOrNoDealXX.cpp, where XX are your initials.

3. A stem-and-leaf plot provides a simple way to visualize the distribution of a set of positive integers. Each integer is split into a "stem" and a "leaf". The leaf is simply the last digit, and the other digits to the left of the leaf form the stem. Here are some examples:

Value	Stem	Leaf
63	6	3
256	25	6
1000	100	0

A single digit number is a leaf with stem equal to 0. In a stem-and-leaf plot, each stem appears at the beginning of its own row and is followed by its leaves. The stems are in ascending order from top to bottom and the leaves are in ascending order from left to right. A colon separates the stem from its leaves. For example, consider the following exam scores: 89, 75, 65, 72, 78, 94, 63, 78, 83, 95, 81, 79 and 86. The stem-and-leaf plot for these scores is:

```
6: 3 5
7: 2 5 8 8 9
8: 1 3 6 9
9: 4 5
```

Write a C++ program that reads a sequence of positive integers until a sentinel value of -1 then displays the stem-and-leaf plot for the given values. Refer to the sample output below.

Sample Runs:

```
Enter positive integers (-1 to quit):
89 75 65 72 78 94 63 78 83 95 81 79 86 -1
```

```
6 : 3 5
7 : 2 5 8 8 9
8 : 1 3 6 9
9 : 4 5
```

```
Enter positive integers (-1 to quit):
123 7 66 23 25 65 67 100 12345 555 -1
```

```
0 : 7
2 : 3 5
6 : 5 6 7
10 : 0
12 : 3
55 : 5
1234 : 5
```

Name the program: StemLeafPlotXX.cpp, where XX are your initials.

4. Given a positive integer M write a C++ program that will print the spiral square matrix which has M rows and M columns. The algorithm to print in a spiral is in the following order:

- DOWN
- OVER (left to right)
- UP
- OVER (right to left)

Error check for values less than 1 and greater than 10. Output to a field width of 5 spaces. Refer to the sample output below.

Sample Run:

Enter the size of the matrix: 5

The spiraled matrix is:

1	16	15	14	13
2	17	24	23	12
3	18	25	22	11
4	19	20	21	10
5	6	7	8	9

Name the program: SpiralMatrixXX.cpp, where XX are your initials.

5. One can find out date of the month when someone was born by asking five questions. Each question asks whether the day is in one of the five sets of numbers below:

1 3 5 7	2 3 6 7	4 5 6 7	8 9 10 11	16 17 18 19
9 11 13 15	10 11 14 15	12 13 14 15	12 13 14 15	20 21 22 23
17 19 21 23	18 19 22 23	20 21 22 23	24 25 26 27	24 25 26 27
25 27 29 31	26 27 30 31	28 29 30 31	28 29 30 31	28 29 30 31
Set1	Set 2	Set 3	Set 4	Set 5

The birthday is the sum of the first numbers in the set where the day appears. For example, if the birthday is 19, it appears in Set 1, Set 2, and Set 5. The first numbers in these three sets are 1, 2, and 16 whose sum is 19. Write a C++ program using arrays that prompts the user to answer whether the day is in Sets 1-5. If the number is in the particular set, the program adds the first number in the sets to calculate the day of the month. An array must be used in the problem. Refer to the sample output below.

Sample Run:

Is your birthday in Set 1?

1	3	5	7
9	11	13	15

17 19 21 23
25 27 29 31

Enter 0 for No and 1 for Yes: 1

Is your birthday in Set 2?

2 3 6 7
10 11 14 15
18 19 22 23
26 27 30 31

Enter 0 for No and 1 for Yes: 1

Is your birthday in Set 3?

4 5 6 7
12 13 14 15
20 21 22 23
28 29 30 31

Enter 0 for No and 1 for Yes: 1

Is your birthday in Set 4?

8 9 10 11
12 13 14 15
24 25 26 27
28 29 30 31

Enter 0 for No and 1 for Yes: 0

Is your birthday in Set 5?

16 17 18 19
20 21 22 23
24 25 26 27
28 29 30 31

Enter 0 for No and 1 for Yes: 0

Your birthday is on day: 7

Name the program: ArrayBirthdateXX.cpp, where XX are your initials.

6 (**). Implement Programming Challenge #16-Chapter 7 (Gaddis/REVEL), *World Series Champions*. Implement the features as stated in the problem. Check validation. Let the user enter the file name from the keyboard. Output should be user friendly.

Name the program: WorldSeriesWinnersXX.cpp, where XX are your initials.

Extra Credit: Implement the problems below for extra credit. See 1436 Grading Guide Sheet for additional grading/submission information. Partial credit will be given.

1. Ugly numbers are numbers whose only prime factors are 2, 3 or 5. The sequence 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, are the first 11 ugly numbers. By convention, 1 is included. Write a C++ program that asks the user for a number n ($n \leq 100000$) and determines if the number entered is an ugly number. Do not use an array. Check for appropriate values. Create a user-defined function. Do not use an array. Refer to the sample output below.

Sample Runs:

Enter a number: 51

51 is not an ugly number. :)

Enter a number: 200

200 is an ugly number. :{

Name the program: UglyXX.cpp, where XX are your initials.

2. Write a C++ program to play the following dice golf game:

The goal is to advance your peg to the tenth hole or beyond. Both players start their peg in the first hole, and taking turns rolling the dice, either advance or move back their pegs, depending on their roll. Each of the possible roll's values are:

Roll	Value	Description
2	+9	Hole in one
3	-2	Sand trap
4	+2	On the green
5	+2	Long drive
6	+1	On the fairway
7	-3	Water hazard
8	+1	On the fairway
9	-2	In the rough
10	-2	Slice
11	+1	Nice chip shot
12	+1	Nice chip shot

It is not possible to move back beyond the first hole; any roll that would take a player backwards past the first hole takes the player's peg to the first hole instead. Any roll that would take a player to the

tenth hole or beyond is a win for that player and the game is considered over. Player 1 always rolls first. Each player will have 10 rolls. The rolls for each player will be randomly generated values from (2-12). For both players output each roll, value, description, and output the final game winner. If there is a tie-state as such. Finally, ask the user if he/she wishes to run the program again. User-defined functions must be used in the program. Output should be user friendly.

Name the program: `DiceGolfXX.cpp`, where XX are your initials.