

CS 102: Fundamentals of Computation

Fall 2020 – Midterm 2

10/28/20 – 7:00 p.m. - 8:25 p.m.
75 min + 10 min to scan/upload = 85 min

[Complete all the information in the box below.]

Name: _____

Student ID: _____

Email: _____@usc.edu

Lecture section (Circle One):

Redekopp
~~1 p.m.~~

Redekopp
2 p.m.

Page	Your score	Max score	Recommended Time
1		-	0 min.
2		14	15 min.
3		18	25 min.
4		18	35 min.
Total		50	75 min.

Each of the 3 problems is described below and at the end of each description is a link/URL to a skeleton file in which you can write your code.

We prefer you upload the finished .cpp files but you may handwrite/annotate the pages below and submit a scanned PDF.

1. A user will enter 3 temperature values. The first 2 will be in units of degrees Fahrenheit while the third **may be in units of Fahrenheit or Centigrade**. To indicate the units of the third temperature input, the user will enter a character **c** or **f** as the fourth and last input. To convert from Centigrade to Fahrenheit can be accomplished using the formula:
$$F = \frac{9}{5}C + 32.$$
 Compare the 3 temperatures (converting the last one to Fahrenheit, if necessary) and output the **SMALLEST** temperature value. If there are 2 or more temperatures tied for the SMALLEST, output either of them (since they are the same number).

Examples:

Input 1 – last input of 'c' indicates third value is in degrees C and must be converted to F
(Note: 37 deg. C = 98.6 deg. F)

99.1 100.4 37 c

Output 1

98.6

Input 2 – Note 40.4 degrees C is 104.72 and thus 97.1 is the smallest.

98.6 97.1 40.4 c

Output 2

97.1

Input 3 – last value is in degrees Fahrenheit and there is a tie for the smallest.

99.3 99.1 99.1 f

Output 3

99.1

The skeleton file on the next page can be downloaded with the link: [fahr.cpp](#).

```
// Compare 3 temperatures in degrees Fahrenheit, though the
// last temperature may be given in degrees Centigrade.
// (see writeup for instructions and requirements)
#include <iostream>
using namespace std;

int main()
{
    // declare necessary variables and complete your
    // code below

    // Your last action should be to output the minimum temperature
    // (in degrees Fahrenheit). Do not print any text, just the
    // temperature and a newline.

    return 0;
}
```

2. Write a program that accepts an arbitrary number of integers and **quits** when either **0** or **13** are entered (because 13 is unlucky). **Total (sum)** the integers that are input according to the following rules:

- **Add** positive numbers that are divisible by 2 (*but not divisible by 3*) to the total
- **Subtract** positive numbers that are divisible by 3 (*but not divisible by 2*) from the total
- **Ignore** (do nothing for) **ALL other numbers** including those that are negative and those divisible by both 2 **AND** 3

Output the sum by itself on a line.

Examples:

Input 1

10 3 0

Output 1: Add 10 (divisible by 2 but not 3) and subtract 3 (divisible by 3 but not 2). **10 - 3 = 7.**

7

Input 2

10 -5 9 14 0

Output 2: Ignore -5, add 10 and 14 (divisible by 2 only) and subtract 9 (divisible by 3 only). **24-9 = 15.**

15

Input 3:

6 20 24 15 3 13 0

Output 3: Ignore 6 and 24 (divisible by 2 and 3), add 20 (divisible by 2 but not 3) and subtract 15 and 3 (divisible by 3 but not 2). We stop at 13 and thus **20 - 18 = 2.**

2

The skeleton file on the next page can be downloaded with the link: [cnt23.cpp](#).

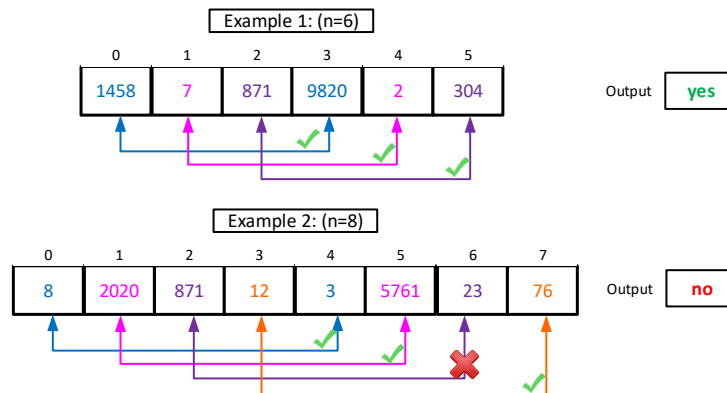
```
// Sum appropriate numbers divisible by 2 or 3
// (see writeup for instructions and requirements)
#include <iostream>
using namespace std;

int main()
{
    // Declare necessary variables and complete your
    // code below.  Final outputs should be made with
    // just the sum on a single line.


    return 0;
}
```

3. **Pairs with the Same Number of Digits.** A user will input an array with an **even** number, **n (max 100)**, of **positive** integers. You should consider the numbers in pairs of two at a time. Pair the **0th** and **middle (n/2) element**, **1st** and **the mid+1 element**, **2nd** and **the mid+2 element**, as shown with the arrows in the diagram below.

Your task is to **check whether the two numbers in each pair have the same number of digits** (e.g. in the 1st example below, the pair 1458 and 9820 each have the same number (four) of digits, the pair 7 and 2 each have the same number of (one) digits, and the pair 871 and 304 each have the same number (three) of digits. **If each pair has the same number of digits**, output **yes**, and otherwise output **no**. Thus, in the array above, the output should be **yes**.



The user will start by entering an **even** integer, **n**, ($0 \leq n \leq 100$ and you may assume **n is even**) and then proceed to enter **n positive** integers. (Note: You **MUST** use integers and not strings). You can then begin to verify if each pair of integers has an equal number of digits. Your solution should define and then call (as many times as needed) the prototyped function: `int numDigits(int val)` which takes a positive integer, **val**, as input and **returns the number of digits in val**.

Examples:

Input 1: (1458 and 9820 each have 4 digits, 7 and 2 each have 1 digit, and 871 and 304 each have 3 digits)

6
1458 7 871 9820 2 304

Output 1:

yes

Input 2: (92 and 7 do NOT have the same number of digits, thus the output should be no.)

4
92 100 7 200

Output 2:

no

The skeleton file on the next page can be downloaded with the link: [samedig.cpp](#).

```

// Check whether pairs of numbers have the same number of digits.
#include <iostream>
#include <cmath> // <cmath> is not necessary but provided if you'd like it
using namespace std;

// Function prototype - Complete the definition below
// Description: Function to count how many decimal digits an integer requires
// Input parameter: val - A non-negative integer
// Return value: number of decimal digits in 'val'
// Example: numDigits(192) should return 3.
int numDigits(int val);

int main()
{
    int vals[100];
    int n;
    cin >> n; // you may assume n is an even number in the range: 0 <= n <= 100

    // write a loop to receive the n integers of input from the keyboard and then complete
    // the remainder of the code to print 'passed' or 'failed', appropriately.
    // You must use the numDigits function by calling it at least once.


    return 0;
}
// See prototype above for description of this function.
int numDigits(int val)
{

}

```