CS102: Fall 2024 – Midterm 2
11/06/24, 100 minutes

Name:_____

Student ID: _____ **Email:** _____**@usc.edu**

I understand I may ONLY access Brightspace, Gradescope Codio, and EdStem (no ChatGPT, Github, or other websites.): _____ **(initials)**

Time Submitted: _____ (if leaving the testing location early)

Lecture section (Circle One): MW 1 PM MW 2PM

Question	1	2	Total
Suggested Time	35 min.	65 min.	100 min.

Instructions:
Use the **question prompts on the following pages** to **fill in your answers on Gradescope..Fa24 MT2. (Link to Gradescope and Codio on Brightspace.. Content.. Exams). You MUST copy/paste (or download/upload) your code from Codio to Gradescope for each problem.** The Gradescope answer form will mark your submission as LATE after **8:40**. Be sure to submit by then.

Common Reference Functions:

- You **MAY NOT** use any functions in the <algorithm> library, but use the reference below.

<iomanip> functions/manipulators:

- `setprecision(n)` – Sets the number of decimal to use when printing a double
- `fixed` – Applies the precision from `setprecision()` to only the decimal portion of the number
- Ex: `cout << fixed << setprecision(3) << 1.12345 << endl; // prints 1.123`

<math> functions:

- `double pow( double base, double exp);` – Returns base^{exp} as a double value
- `double abs (double x);` returns the absolute value of the input, x
- `double sqrt(double x);` – Returns the square root of the input, x

<string> functions (for the `string` type):

- `.size()` – Returns the number of characters in the string (does not count a null character).
- `.substr(int first)` – Returns the substring starting at index `first` through the end of the string.
- `.substr(int first, int num)` – Returns the substring of `first+num-1` characters starting at index `first`.

Examples:

```
string s1 = "abcd";
s1.size();      // returns 4
s1.substr(1);   // returns "bcd"
s1.substr(1,2); // returns "bc"
```

Question 1 - Increasing Sequence

Open `incseq.cpp` and implement the program before copying the code to Gradescope.

In this program, the user will enter a sequence of **positive** numbers and your code should check if the sequence is in increasing order (each one **strictly larger** than the last). If the entire sequence of positive numbers is increasing (ascending), output the **SUM** of all the **POSITIVE** integers. Otherwise, just output the word: **no**. The user may type in ANY AMOUNT of positive input values (no limit on how many numbers may be input) but your program should stop when the user enters a **negative** number (but ignore that negative number). You may also end the program as soon as a positive input would cause the output to be **no**.

Example 1: Given the input: `2 50 100 -3` output **`152`** since the sequence of positive numbers are all increasing.

Example 2: Given the input: `2 4 6 8 1001 1003 -1` output **`2024`** since the sequence of positive numbers are all increasing.

Example 3: Given the input: `50 20 70 80 -1` output **no** since `20` is not larger than `50`. You may output no and end the program immediately after the input `20` is received or wait for a negative input.

Requirements and Assumptions

- **NO <u>ARRAYS</u> ARE NEEDED NOR MAY THEY BE USED.**
- To receive credit, you **MUST pass the automated, Codio tests**. No partial credit will be given even if the issue was a minor output format error.
- You may assume correctly formatted input values (i.e. no text, only numbers) and that the sequence is ended by a negative number.
- You may ONLY use the `iostream` library. NO other libraries may be included or used.
- Solutions containing EXTREMELY excessive/unnecessary code may not receive credit.
- A skeleton is provided on the next page for annotation or use in case your laptop dies. In that scenario, write your code by hand on the next page and when the exam is finished, hand it to the head TA and clearly state that your laptop died and you need us to grade the paper code.

```
// The question prompt itself is written on the paper exam. Refer to it.
// No other #include statements may be added
#include <iostream>
using namespace std;

int main() {









    return 0;
}
```

Question 2 - Matching Evens

Open `match-evens.cpp` in Codio and implement the program before copying the code to Gradescope.

In this program, the user will first enter a positive integer, **n**, between 1 and `100` followed by a sequence of **n integers (which could be positive or negative).** You should **ignore** all the **odd** numbers but verify if ALL **even** numbers in the sequence have **one or more matching (equal) numbers**, and output **`yes`** if so, and **no**, otherwise.

Example 1: Given the input:

```
6
2 -1 2 3 8 10
```

output **no** since the even numbers `8` and `10` do NOT have at least one other equal match.

Example 2: Given the input:

```
7
-2048 17 8 32768 32768 -2048 8
```

output **`yes`** since all the even numbers (`-2048, 8, 32768`) have at least one equal match.

Example 3: Given the input:

```
8
4 -1 4 3 4 4 -5 4
```

output **`yes`** since all the even numbers (i.e. `4`) have an equal match.

Example 4: Given the input:

```
1
3
```

output **`yes`** since technically all (ZERO) even numbers have an equal match.

Requirements and Assumptions

- **We have declared an array for you and provided the code to handle the input.**
- No OTHER arrays may be declared or used beyond the one declared in the skeleton.
- You may assume correctly formatted input values (i.e. no text, only numbers).
- You may ONLY use items from the `iostream` library. NO other libraries may be included.
- Solutions containing EXTREMELY excessive/unnecessary code may not receive credit.
- A skeleton is provided on the next page for annotation or use in case your laptop dies. In that scenario, write your code by hand on the next page and when the exam is finished, hand it to the head TA and clearly state that your laptop died and you need us to grade the paper code.

```
#include <iostream>
using namespace std;

int main() {
    int data[100], n;
    // Read in the size of the portion of the array to be used
    cin >> n;
    // Then read in n integers
    for(int i = 0; i < n; i++) {
        cin >> data[i];
    }

    return 0;
}
```