NYU, Tandon School of Engineering
Bridge to CS — Spring 2021

Exam #2

Thursday, May 27, 2021

- This exam has 6 questions, with 100 points total.

- **You should submit your answers in the Gradescope platform (not on NYU Classes).**

- You have **two hours**.

- **It is your responsibility to take the time for the exam** (You may use a physical timer, or an online timer: https://vclock.com/set-timer-for-2-hours/).
  Make sure to upload the files with your answers to gradescope BEFORE the time is up, while still being monitored by ProctorU.
  We will not accept any late submissions.

- In total, you should upload 3 '.cpp' files:
  - One '.cpp' file for questions 1-4.
    Write your answer as one long comment (/* … */).
    Name this file 'YourNetID_q1to4.cpp'.
  - One '.cpp' file for question 5, containing your code.
    Name this file 'YourNetID_q5.cpp'.
  - One '.cpp' file for question 6, containing your code.
    Name this file 'YourNetID_q6.cpp'.

- **Write your name, and netID at the head of each file.**

- This is a closed-book exam. However, you are allowed to use:
  - CLion or Visual-Studio. You should create a new project and work ONLY in it.
  - Two sheets of scratch paper.
  - Scientific Calculator

  Besides that, no additional resources (of any form) are allowed.

- You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.

- Read every question completely before answering it.
  Note that there are 2 programming problems at the end.
  Be sure to allow enough time for these questions

Part I – Theoretical:

- *You should submit your answers to all questions in this part (questions 1-4) in* ***one*** *'.cpp' file. Write your answers as one long comment (`/* … */`).*
  Name this file 'YourNetID_q1to4.cpp'.

- *For questions in this part, try to find a way to use regular symbols.*
  For example, instead of writing a^b you could write a^b, instead of writing $\Theta(n)$, you could write theta(n), instead of writing $\binom{n}{k}$ you could write C(n, k), etc.
  Alternatively, you could also make a note, at the beginning of your answer, stating what symbol you used to indicate a specific mathematical notation.

Question 1 (14 points)

Use mathematical induction to prove that for any positive integer n, 6 evenly divides $7^n - 1$.
Note: 6 evenly divides an integer num, if and only if num is a multiple of 6.
For example, 6 evenly divides 18 (18 is a multiple of 6).

Question 2 (12 points)

You are creating a 4-digit pin code. How many choices are there where exactly one digit appears more than once?
Explain your answer.

Question 3 (15 points)

Two fair dice are rolled. Let *X* be the random variable: if the values in the dice are d_1 and d_2, then $X = \max(d_1, d_2)$
For example:

- if when rolling the dice, you get 4 and 3, *X* would be 4
- if when rolling the dice, you get 2 and 2, *X* would be 2.

a. Find the distribution of *X*. That is, for each possible value of *X*, say what is the probability *X* would get that value.
b. What is *E(X)*? That is, find the expected value of *X*.
Explain your answers.

Question 4 (14 points)
Analyze its running time of **`func1`** and **`func2`**.
Explain your answers.

Note: Give your answers in terms of asymptotic order. That is, $T(n) = \Theta(n^2)$, or $T(n) = \Theta(\sqrt{n})$, etc.

```
void func1(int n){
    int count;
    int i, j;

    count = 0;
    for(i = 1; i <= n; i += 1){
        for(j = 1; j <= n; j += 1){
            count += 1;
        }
    }

    for(i = 1; i <= count; i += 1){
        cout<<”func1 is awesome!”<<endl;
    }
}


void func2(int n){
    int sum;
    int i, j;

    sum = 0;
    for(i = 1; i <= n; i += 1){
        for(j = 1; j <= n; j += 1){
            sum += i;
        }
    }

    for(i = 1; i <= sum; i += 1){
        cout<<”func2 is awesome!”<<endl;
    }
}
```

Part II – Coding:

- *Each question in this part (questions 5-6), should be submitted as a '.cpp' file.*
- *Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose most suitable control statements, etc.*
- *In all questions, you may assume that the user enters inputs as they are asked. For example, if the program expects a positive integer, you may assume that user will enter positive integers.*
- *No need to document your code. However, you may add comments if you think they are needed for clarity.*

Question 5 (30 points)

Given a list of numbers $\mathrm{lst} =< a_1, a_2, \dots, a_n >$ where all the numbers are taken from the range $\{1, 2, 3, \dots, \mathrm{n}\}$. We say that lst is a **symmetric-distribution** if all the following holds:

- (number of 1's in lst) = (number of n's in lst)
- (number of 2's in lst) = (number of $(\mathrm{n} - 1)$'s in lst)
- (number of 3's in lst) = (number of $(n - 2)$'s in lst)
- Etc.

For example, if $\mathrm{n} = 9$, the list $< 9, 7, 5, 5, 3, 5, 1, 9, 1 >$ is a symmetric-distribution, since all the values are in the range 1-9 and, there are:

- 2 times 1, and also 2 times 9
- 0 times 2, and also 0 times 8
- 1 time 3, and also 1 time 7
- 0 times 4, and also 0 times 6
- 3 times 5 (and also 3 times 5)

Implement the function:

```
bool isSymmetricDistribution(int arr[], int n)
```

This function gets an array of integers `arr` and its logical size `n`. All elements in `arr` are in the range $\{1, 2, 3, \dots, \mathrm{n}\}$. That is, `arr` contains only integers in the range 1-n. When called, it should return `true` if `arr` is a symmetric-distribution or `false` otherwise.

For example, if `arr=[9, 7, 5, 5, 3, 5, 1, 9, 1]`, the call `isSymmetricDistribution(arr, 9)`, should return `true`.

Implementation requirements:

1. Your function should run in $\theta(n)$.
2. In this question you are not allowed to use any library besides iostream. That is, you are not allowed to use `cmath`, `vector`, `string`, etc.

Hint: Try to think how to use that fact that the numbers are all in the range 1-n.

Question 6 (15 points)
Consider the following definitions:

- A **palindrome** is a sequence, which reads the same backward or forward.
  For example:
  - *[3, 3, 12, 12, 3, 3]* is a palindrome
  - *[4, 5, 5, 4]* is a palindrome
  - *[3, 3, 12, 12, 9, 9, 3, 3]* is **not** a palindrome
- A **doubles-palindrome** is a palindrome in which the first and the second numbers are the same, the third and the fourth numbers are the same, the fifth and sixth are the same, etc.
  For example:
  - *[3, 3, 12, 12, 3, 3]* is a doubles-palindrome
  - *[4, 5, 5, 4]* is **not** a doubles-palindrome
  - *[3, 3, 12, 12, 9, 9, 3, 3]* is **not** a doubles-palindrome

Give a **recursive** implementation for:

```
bool isDoublesPalindrome(int seq[], int n)
```

The function is given `seq`, an array containing a sequence of integers, and its logical size, `n`. When called, it should return `true`, if `seq` is a doubles-palindrome, or `false` otherwise.

Notes:
1. You don't need to write a `main()` program.
2. Your function **must be recursive**.