NYU, Tandon School of Engineering
Bridge to CS — Summer 2020

Exam #2

Thursday, September 3, 2020

- 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.
  You may also use two sheets of scratch paper.
  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.

- Calculators are **not** allowed.

- 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)

Let a_n be the Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, ...).

That is: $a_n = \begin{cases} 1 & (n = 1 \lor n = 2) \\ a_{n-1} + a_{n-2} & n > 2 \end{cases}$

Use mathematical induction to show that for every positive integer: $a_n \leq 2^{n-1}$

Question 2 (12 points)

A **ternary string** is a sequence of digits, where each digit is either 0, 1, or 2.
For example "011202" is a ternary string of length 6.

How many ternary strings of length 6 have the same number of 1s and 2s?
Explain your answer.

Question 3 (15 points)

A blue die and a red die are thrown in a game. If the sum of the two numbers is 7 or 11, the player wins \$10. If the sum of the two numbers is 12, then the player wins \$20. In all other cases, the player loses a dollar.
Let *X* be the random variable that denotes the winnings of the player in one game.

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.

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

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

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

    arr = new int[n];
    arr[0] = 1;
    for (i = 1; i < n; i++)
        arr[i] = arr[i-1] + 2;

    count = 0;
    for (i = 0; i < n; i++)
        for (j = 1; j <= arr[i]; j++)
            count++;

    delete []arr;
    return count;
}
```

```
int func2(int n){
    int i, j;
    int count;
    int* arr;

    arr = new int[n];
    arr[0] = 1;
    for (i = 1; i < n; i++)
        arr[i] = arr[i-1] * 2;

    count = 0;
    for (i = 0; i < n; i++)
        for (j = 1; j <= arr[i]; j++)
            count++;

    delete []arr;
    return count;
}
```

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)

The ***median*** of a finite list of numbers is the "middle" number, when those numbers are listed in order from smallest to greatest.

- If there is an odd number of numbers, the middle one is picked.
  For example, in the data set: <*3, 5, 3, 7, 1, 7, 6*>, the median is 5 (since 5 is the middle element in the sorted sequence: <*1, 3, 3,* ***5****, 6, 7, 7*>).
- If there is an even number of numbers, then there is no single middle value; the median is then defined to be the average of the two middle elements.
  For example, in the data set: <*4, 1, 6, 2, 7, 3, 2, 8*>, the median is 3.5 (since 3.5 is the average of the middle two elements in the sorted sequence:
  <*1, 2, 2,* ***3, 4****, 6, 7, 8*>).

Implement the function:
`double findMedian(int arr[], int n)`

This function gets an array of positive integers `arr` and its logical size `n`. All elements in `arr` are in the range *{1, 2, … , n}*. That is, `arr` contains only positive integers in the range 1-n.
When called, it should return the median of `arr`.

For example, if `arr=[3, 5, 3, 7, 1, 7, 6]`, the call `findMedian(arr, 7)`, should return `5`.

Note: For simplicity you may assume that `arr` contains odd number of elements. That is, assume that `n` is odd.

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: To meet the time requirement, you would **not** want to actually sort the elements, since sorting an arbitrary set of n integers requires more than $\theta(n)$. Instead, try to think how to use that fact that the numbers are all in the range 1-n.

Question 6 (15 points)

Give a **recursive** implementation for:

```
int findFirstPosition(int arr[], int arrSize, int elem)
```

The function is given `arr`, an array containing integers, and its logical size, `arrSize`. In addition, the function is given an integer `elem`.
When called, it should return the index where `elem` **shows first** in the array `arr`. If `elem` does not show at all in arr, the function should return -1.

For example, if `arr = [2, 15, 3, 8, 3, 10, 6, 23, 12, 32]`:

- The call `findFirstPosition(arr, 10, 3)` should return 2 (since 2 is the index where 3 appears first).
- The call `findFirstPosition(arr, 10, 7)` should return -1 (since 7 is not an element in `arr`).

Notes:

1. You don't need to write a `main()` program.
2. Make sure that is `elem` shows in `arr` more than once, your function would return the index where it appears **first**.
3. Your function **must be recursive**.