NYU, Tandon School of Engineering
Extended Bridge to CS — Winter 2020

Exam #2

Thursday, March 26, 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.
  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 for section (a), and the answer to section (b) typed as a comment.
    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.

- 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)
Use **mathematical induction** to show that 2 divides n^2+n whenever n is a positive integer.

Question 2 (12 points)
A class of *5n* students, with *3n* boys and *2n* girls, wants to select *n* students to write a report. How many ways are there to select the *n* students, so that at least one girl is selected?
Explain your answer.

Question 3 (15 points)
A player spins two spinners. The outcome of each spinner is 1, 2, or 3. Each outcome is equally likely.
Let *X* be the random variable that denotes the maximum of the two numbers on the spinners.

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.

```
int func1(int n){
    int i, j;
    int sum;

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

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

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

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

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

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

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

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)

a. Implement the following function:

```
void makeDifferenceArray(int* srtArr1, int n1, int* srtArr2, int n2,
                         int*& outDifferenceArr, int& outN)
```

This function takes two base addresses of arrays of integers: `srtArr1` and `srtArr2`, and their respective logical sizes: `n1` and `n2`. The elements in each array come in a **sorted** (strictly ascending) order.
When called, it should create an array, containing all the **numbers that are in `srtArr1` and not in `srtArr2`**. This array should be returned by using the two output parameters: `outDifferenceArr` and `outN` to update the base address of the created array, and its logical size.
Note that these output parameters use call by reference (not pointers) to update the values in the scope of the caller of the function.

For example, if `srtArr1=[1, 2, 3, 5, 7, 8]`, and `srtArr2=[2, 5, 6, 9]`, the call to `makeDifferenceArray` with these two arrays, should create the array `[1, 3, 7, 8]` and use the output parameters to return its base address and its logical size.

Runtime requirement:
Your function should **run in worst-case linear time** $(\Theta(n_1 + n_2))$.

Implementation requirements:

1. You must implement the function with the prototype as given above. That is, you are not allowed to change the header line of the function.
2. The function should not change the contents of its input arrays.

b. You are given the following program, that calls the function described in section (a):

```
1.  int main(){
2.      int srtArr1[6] = {1, 2, 3, 5, 7, 8};
3.      int srtArr2[4] = {2, 5, 6, 9};
4.      int* differenceArr;
5.      int differenceArrSize;
6.
7.      makeDifferenceArray(srtArr1, 6, srtArr2, 4, ____, ____);
8.
9.      cout<<"srtArr1:  ";
10.     printArray(srtArr1, 6);
11.
12.     cout<<"srtArr2:  ";
13.     printArray(srtArr2, 4);
14.
15.     cout<<"differenceArr:  ";
16.     printArray(differenceArr, differenceArrSize);
17.
18.     delete []differenceArr;
19.     return 0;
20. }
21.
22. void printArray(int arr[], int arrSize){
23.     for(int i = 0; i < arrSize; i++)
24.         cout<<arr[i]<<"  ";
25.     cout<<endl;
26. }
```

Complete line 7 of the program above, so when executed, it would print:

```
srtArr1: 1  2  3  5  7  8
srtArr2: 2  5  6  9
differenceArr: 1  3  7  8
```

Question 6 (15 points)

A **palindrome** is a sequence, which reads the same backward or forward. For example, *[3, 12, 12, 3]*, *[-2, 4, 5, 4, -2]*, and *[1, 1, 2, 3, 3, 2, 1, 1]* are all palindromes.

Give a **recursive** implementation for:

```
bool isPalindrome(int seq[], int seqSize)
```

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

Notes:

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