Exam 2

Thursday, March 14, 2024

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

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

- 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:
  - Visual-Studio, Visual Studio Code (VSCode), Xcode, CLion. You should create a new project and work ONLY in it.
  - Two sheets of scratch paper.
  - Scientific Calculator (Physical or Operating System's Provided One).

  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 (13 points)

Use mathematical induction to prove that 3 divides $n^3 + 2n$ whenever n is a positive integer.

Hint: you may want to use the formula: $(a + b)^3 = a^3 + 3\,a^2b + 3a\,b^2 + b^3$.

Question 2 (16 points)

a) How many bit strings of length seven either begin with two 0s or end with three 1s? **Explain your answer.**

b) How many ways are there for 5 women and 4 men to stand in a line so that no two men stand next to each other? **Explain your answer.**

Question 3 (18 points)

a) Find the probability of at most one success when n independent Bernoulli trials are carried out with probability of success p? **Explain your answer.**

a) You have seven cards, numbered 3 through 9, and you pick one at random. If you pick a card with a prime number, you get 1 point; if you pick a card with a composite number (that is, a number which not prime), you lose 1 point. Find the expected value of the number of points you get. **Explain your answer.**

Question 4 (18 points)

Analyze its running time of `function1` and `function2`.

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 function1(int n){
    int i, j;
    int sum = 0;

    i = 1;
    while (i <= n){
        for (j = 1; j <= i; j++)
            sum += j;
        i *= 2;
    }

    for (i = 1; i <= n; i *= 2){
        for (j = 1; j <= i; j++){
            sum += i;
        }
    }

    return sum;
}


int function2(int n){
    int i, j;
    int sum = 0;

    for (i = 1; i <= n/2; i += 2)
        sum += 1;

    for (i = 1; i <= n; i *= 2){
        j = i;
        while (j > 1){
            sum += 1;
            j /= 2;
        }
    }


    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 (18 points)

Give a **recursive** C++ implementation for the function:

```
bool moreEvens(string S)
```

The above function is given a **non-empty** numeric string **S** that will **only** contain **decimal character digits [`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`].** When this `moreEvens` function is called, it should **return true if total number of even digit characters** in string S **is more than** the **total number of odd digit characters** in string S. Otherwise it will **return false. Even digit characters are [`0`, `2`, `4`, `6`, `8`] and odd digit characters are [`1`, `3`, `5`, `7`, `9`].**

For example, if **S = "123456"**, after calling `moreEvens(S)`, this function should return **false**.

For example, if **S = "01234526"**, after calling `moreEvens(S)`, this function should return **true**.

For example, if **S = "2"**, after calling `moreEvens(S)`, this function should return **true**.

For example, if **S = "22334512378"**, after calling `moreEvens(S)`, this function should return **false**.

Implementation requirements:

- Your function should run in **worst case linear time**. That is, it should run in $\boldsymbol{\theta(n)}$ where n = length of the string S.
- Your function **must be recursive**.
- If you need, you may use additional/helper function with additional parameters. If your additional/helper function is recursive and you call that function from the `moreEvens` function, it will satisfy the requirement of being recursive.
- You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.

Note: You don't need to write a `main()` function.

Question 6 (17 points):

In this question, you should write a program that reads a sequence of positive integers (each integer will consist of 3 to 5 decimal digits).

The input would be entered as a non-empty sequence of lines, where each line would contain a positive integer (each integer will consist of 3 to 5 decimal digits) and a -1 will indicate the end of the input.

After reading the input, the program would print all the 3 digits integers (according to the insertion order of all the 3 digits integers), then print all the 4 digits integers (accordingly to the insertion order of all the 4 digits integers), and then print all the 5 digits integers (according to the insertion order of all the 5 digits integers). Your program should also print the total summation of all the digits in all the 3 digits integers, then print the total summation of all the digits in all the 4 digits integers, then print the total summation of all the digits in all the 5 digits integers.

Your program should interact with the user **exactly** the same way, as demonstrated in the following two executions:

Execution example 1:

```
Please enter a sequence of positive integers (each integer will have
at least 3 decimal digits and at most 5 decimal digits and first
digit of each integer won't be 0), each one in a separate line. End
your sequence by typing -1:
123
4567
103
12345
234
101
1111
345
10101
-1
123
103
234
101
345
4567
1111
12345
10101
Summation of all the digits in all the 3 digits integers: 33
Summation of all the digits in all the 4 digits integers: 26
Summation of all the digits in all the 5 digits integers: 18
```

Execution example 2:

```
Please enter a sequence of positive integers (each integer will have
at least 3 decimal digits and at most 5 decimal digits and first
digit of each integer won't be 0), each one in a separate line. End
your sequence by typing -1:
98999
100
4000
100
1000
10000
103
2020
99999
9999
-1
100
100
103
4000
1000
2020
9999
98999
10000
99999
Summation of all the digits in all the 3 digits integers: 6
Summation of all the digits in all the 4 digits integers: 45
Summation of all the digits in all the 5 digits integers: 90
```

Implementation requirements:

- Your program should ignore the inputted -1 that was used to indicate the end of input.
- Make sure to **design your program best**. In particular, break your implementation to functions.
- You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.