Exam 2

Thursday, September 22, 2022

- 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. 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 for all integers $n \geq 0$,
$1 + 3 + 9 + 27 + \cdots + 3^n = (3^{n+1} - 1)/2$.

Question 2 (16 points)

a) A class consists of **20** sophomores and **15** freshmen. The class needs to form a committee of size five. How many committees are possible if the committee must have three sophomores and two freshmen? **Explain your answer.**

b) In this question, consider all bit strings of length **12**. How many of those bit strings begin with **11** and end with **10**? **Explain your answer.**

Question 3 (18 points)

a) A bowl has eight ping pong balls numbered **1, 2, 2, 3, 4, 5, 5, 5**. You pick a ball at random. What is the probability that the number on the ball drawn is greater than or equal to **3**. **Explain your answer.**

b) A red and a green die are rolled. What is the probability of getting a sum of six, given that the number on the green die is odd? **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;


    for (i = 1; i <= n; i *= 3)
        for (j = 1; j <= n; j++)
            sum += (i+j);

    if(n%2 == 0){
        for (i = 1; i <= n; i *= 2)
            for (j = 1; j <= n; j++)
                sum += (i+j);
    }

    return sum;
}


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

    for (k = 1; k <= n; k += 1){
        for (i = 1; i <= n; i *= 3){
            j = i;
            while (j > 1){
                sum += 1;
                j /= 3;
            }
         }
     }

    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 (20 points)

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

```
int findIndex(int S[], int x, int left, int right)
```

The above function is given an integer array **S** that will contain **unique integers in the sorted order (ascending order),** an integer **x,** an integer **left** that will indicate the index of the first element in array **S**, and an integer **right** which will indicate the index of the last element in array **S**. When this `findIndex` function is called, it should **return the index of the first appearance of element x in array S or -1 if element x does not appear in array S.**

For example, if **S = {10, 15, 20, 25, 28, 40, 55, 90}**, after calling `findIndex(S, 39, 0, 7)`, this function should return **-1**.

For example, if **S = {-1, 5, 17, 45, 66, 80, 90, 95, 144, 356}**, after calling `findIndex(S, 95, 0, 9)`, this function should return **7**.

For example, if **S = {-87, 10, 18, 19, 30, 40, 49, 74, 102, 178}**, after calling `findIndex(S, 18, 0, 9)`, this function should return **2**.

For example, if **S = {9, 14, 38, 55, 66, 71, 78, 84, 129, 178, 234}**, after calling `findIndex(S, 11, 0, 10)`, this function should return **-1**.

Implementation requirements:

- Your function should run in **worst case Logarithmic time**. That is, it should run in $\boldsymbol{\theta(logn)}$ where n = logical size of the array S or n = (right-left)+1.
- Your function **must be 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 (15 points)

In this question, you should write a program that reads a sequence of strings (each string will consist of lowercase English letters and/or special characters **#, @, $, !, &**) and removes all the special characters from each string and then prints the strings (after removal of special characters) according to the insertion order. If a string becomes empty after removing special characters, you don't need to print that string. At the end, you should print how many strings have become empty strings after removal of the special characters and you should also print how many strings don't contain any special characters at all. That is, the program will remove special characters from each of the input strings and print those strings according to the insertion orders, and then print how many of the input strings become empty after removal of special characters and then also print how many of the input strings don't contain any special characters at all.

The input would be entered as a non-empty sequence of lines, where each line would contain a single string (each string will consist of lowercase English letters and/or special characters **#, @, $, !, &**), and an empty string will indicate the end of the input.

After reading the input, the program would remove special characters from each of the strings and then print those strings (after removal of the special characters) maintaining the insertion order, followed by a number that will indicate how many strings in the input sequence are consisted of only special characters and then followed by a number that will indicate how many strings in the input sequence don't contain any special characters at all. If a string in the input sequence becomes empty after removal of special characters, you don't need to print that empty string. Your program should ignore the inputted empty string that was used to indicate the end of input.

Your program should interact with the user **exactly**, as demonstrated below:

Example 1:

```
Please enter a non-empty sequence of Strings. Each string should
be in a separate line and consists of only lowercase English
letters and/or special characters @, #, $, !, &. To indicate the
end of the input, enter an empty string in one line.
abc@#def$!&ghi
!&@#$
&&&@#$
utwhjhyruyr
hkhaaoik@#uiu&@!
#kkmnjk$@#

abcdefghi
utwhjhyruyr
hkhaaoikuiu
kkmnjk
Number of Strings in the input sequence that contain only special
characters:  2
Number of Strings in the input sequence that contain only lowercase
English letters:  1
```

Example 2:

```
Please enter a non-empty sequence of Strings. Each string should
be in a separate line and consists of only lowercase English
letters and/or special characters @, #, $, !, &. To indicate the
end of the input, enter an empty string in one line.
```

abcxyz
#bauaguiw!@#$&hdhshsyuh@uequ
@@@$$!!$$@$
ywiuiqghsdefcmn
#j@j@i@p#
abcxyz
!@#$&&$#@!
mkjhyuioperdswqa
abcdeftqtyqraf#
wer!!!pi###uyteuoa&
@@@@@@

abcxyz
bauaguiwhdhshsyuhuequ
ywiuiqghsdefcmn
abcxyz
mkjhyuioperdswqa
ytqrytu
abcdeftqtyqraf
werpiuyteuoa

```
Number of Strings in the input sequence that contain only special
characters:  3
Number of Strings in the input sequence that contain only lowercase
English letters:  3
```

Notes:

1. Your program should ignore the inputted empty string that was used to indicate the end of input.
2. Make sure to **design your program best**. In particular, break your implementation to functions.
3. You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.
4. You need to consider only 5 special characters for this question. These are **@, #, $, !, &**.