Exam 2

Thursday, April 4, 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 $2n + 3 \leq 2^n$ whenever n is a positive integer and $n \geq 4$.

Question 2 (16 points)

a) How many strings of four decimal digits do not contain the same digit more than once? Note that first character of these strings can be '0'. **Explain your answer.**

b) A multiple-choice test contains six questions. There are three possible answers for each question. In how many ways can a student answer the questions on the test if the student can leave answers blank? **Explain your answer.**

Question 3 (18 points)

a) A group of seven women and seven men are in a room. A committee of three is chosen at random. Find the probability that the committee consists only of women? **Explain your answer.**

b) Suppose you flip a biased coin (where probability of getting head is 3/5 and probability of getting tail is 2/5) **8 times**. Find the probability of getting **at least 7 heads** out of these **8 flips**. **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 *= 2)
        for (j = 1; j <= i; j++)
            sum += (i+j);

    i = 1;

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


    return sum;
}
```

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

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

    sum = 0;

    for (i = 1; i <= n; i *= 2){
        j = 1;
        while (j <= i){
            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 (17 points)

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

```
void print_up_down(unsigned int n)
```

The above function is given an unsigned **integer n** and **n** is greater than **0** **(n > 0).** When this **`print_up_down`** function is called, it should **print the (2 *n -1) integers as follows:**

1
2
3
.
.
.
n
n-1
n-2
.
.
.
3
2
1

Implementation requirements:

- Your function should run in **worst case linear time**. That is, it should run in $\boldsymbol{\theta(n)}$.
- 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 **`print_up_down`** 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.

For example, if n = 5 and we call **print_up_down**(n)function, this function should print as follows.

```
1
2
3
4
5
4
3
2
1
```

For example, if n = 1 and we call **print_up_down**(n)function, this function should print as follows.

```
1
```

For example, if n = 10 and we call **print_up_down**(n)function, this function should print as follows.

```
1
2
3
4
5
6
7
8
9
10
9
8
7
6
5
4
3
2
1
```

Question 6 (18 points):
Give a C++ implementation for the function:

```
void removeOdds(vector<int>& Vector);
```

The above function is given an address to an integer vector **Vector** (**type vector <int>**) that will contain the positive integers. When this `removeOdds` function is called, it should remove all the odd positive integers from **Vector** (**type** vector<int>) and keep only the even positive integers in **Vector** (**type** vector<int>). Note that after removing the odd positive integers from **Vector** (**type** vector<int>), the order of the remaining even positive integers in **Vector** (**type** vector<int>) does not matter.

For example, if type of **Vector** variable is vector<int>, and this is initialized as vector<int> **Vector {100, 75, 20, 15, 5, 2, 6},** after calling `removeOdds(Vector)`, **Vector** could be `{100, 20, 2, 6}`. After removing odd integers, order of the even integers in **Vector** does not matter. After removing, `another of the other possible values of` **Vector** `could be {100, 6, 20, 2}.`

For example, if type of **Vector** variable is vector<int>, and this is initialized as vector<int> **Vector {5, 0, 75, 22, 19, 15, 21, 16},** after calling `removeOdds(Vector)`, **Vector** could be `{0, 22, 16}`. After removing odd integers, order of the even integers in **Vector** does not matter. After removing, `another of the other possible values of` **Vector** `could be {16, 0, 22}.`

For example, if type of **Vector** variable is vector<int>, and this is initialized as vector<int> **Vector {2, 0, 1, 22, 34, 53, 18, 16},** after calling `removeOdds(Vector)`, **Vector** could be `{2, 0, 22, 34, 18, 16}`. After removing odd integers, order of the even integers in **Vector** does not matter. After removing, `another of the other possible values of` **Vector** `could be {2, 0, 16, 22, 34, 18}.`

Implementation requirements:

- Your function should run in $\boldsymbol{\theta(n)}$ time or in amortized $\boldsymbol{\theta(n)}$ time where n = initial size of the vector **Vector**. For Simplicity, you can assume that amortized $\boldsymbol{\theta(1)}$ is same as $\boldsymbol{\theta(1)}$ or amortized $\boldsymbol{\theta(n)}$ is same as $\boldsymbol{\theta(n)}$.
- You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.
- For this question, you can assume that 0 is a positive even integer.
- You may use resize() function of the vector class for resizing the **Vector** (**type** vector<int>). Calling format is Vector.resize(new_size) where new_size is the new size of the **Vector** and resize() function runs in amortized $\boldsymbol{\theta(new\_size)}$.
- Your function should use $\boldsymbol{\theta(1)}$ additional memory, that is, you should not create a new vector/array in your function. You need to modify the parameter **Vector** (**type** vector<int>).

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