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

Exam #2

Thursday, September 24, 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 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.

- 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)
Use **mathematical induction** to show that 3 divides $2^{2n+1} + 1$ whenever n is a positive integer.

Question 2 (12 points)
Peter has 100 balls:
- 30 red balls numbered 1, 2, 3, …, 30
- 70 green balls numbered 1, 2, 3, …, 70.

In how many ways can he pick 20 balls, such that there will be exactly k red balls?
Explain your answer.

Question 3 (15 points)
At a horse race, your horse has a probability of $\frac{1}{20}$ of coming in first place, a probability of $\frac{1}{10}$ of coming in second place, and a probability of $\frac{1}{4}$ of coming in third place.
You need to pay $1,000 to enter the race, and then first place pays $4,500 to the winner, second place $3,500 and third place $1,500.

Let X be the random variable that denotes the value of your **net winnings**.

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 count;

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

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

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

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

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

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

    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)

In this question we represent a positive integer in its base-2 representation, and store it in an array of `int`s. The elements of the array would be the bits of the number, in the order from most-significant to least-significant.
For example, since $38 = (100110)_2$, 38 would be represented with the following 6-element array: [1, 0, 0, 1, 1, 0].

a. Implement the following function:
```
void intToBitsArr(int num, int** outBitsArr, int& outBitsArrSize)
```

When this function is called with a positive integer `num`, it should create an array, containing the bits of its base-2 representation, as described above.
This array should be returned by using the two output parameters: `outBitsArr` and `outBitsArrSize` to update the base address of the created array, and its logical size. Note that these output parameters use different methods to update the values in the scope of the caller of the function. `outBitsArr` uses a pointer, where `outBitsArrSize` uses call by reference.

For example, the call to `intToBitsArr` with `num=38`, should create the array `[1, 0, 0, 1, 1, 0]` and use the output parameters to return its base address and its logical size.

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. Aim for the best asymptotic runtime.

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

```
1.  int main(){
2.      int* bitsArr;
3.      int bitsArrSize;
4.
5.      intToBitsArr(38, ______, ______);
6.
7.      cout<<"bitsArr:  ";
8.      printArray(bitsArr, bitsArrSize);
9.
10.     delete []bitsArr;
11.     return 0;
12. }
13.
14. void printArray(int arr[], int arrSize){
15.     for(int i = 0; i < arrSize; i++)
16.         cout<<arr[i]<<" ";
17.     cout<<endl;
18. }
```

Complete line 5 of the program above, so when executed, it would print:

```
bitsArr: 1 0 0 1 1 0
```

Question 6 (15 points)
In this question we (again) represent a positive integer in its base-2 representation, and store it in an array of `int`s. The elements of the array would be the bits of the number, in the order from most-significant to least-significant.
For example, since $38 = (100110)_2$, 38 would be represented with the following 6-element array: [1, 0, 0, 1, 1, 0].

Give a **recursive** implementation for:

```
int bitsArrToInt(int* bitsArr, int bitsArrSize)
```

This function is given an array as an input. That is, `bitsArr` is the base address of the array, and `bitsArrSize` is its logical size. The array `bitsArr` contains the bits (0s and 1s) of a positive integer in its base-2 representation, as described above.
When this function is called, it should return the positive integer that is represented in `bitsArr`.

For example, if `bitsArr` = `[1, 0, 0, 1, 1, 0]`, the call `bitsArrToInt(bitsArr, 6)` should return 38.

Notes:
1. You don't need to write a `main()` program.
2. Your function **MUST BE RECURSIVE**.