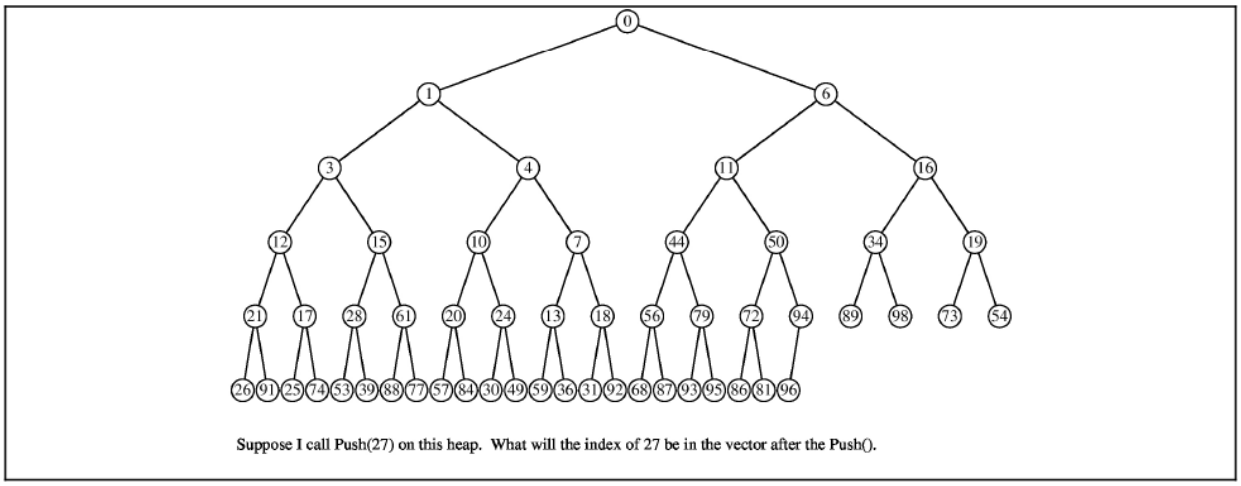
CS302 Midterm 2020

The exam was on Canvas, and I used question banks for a lot of questions, so I won't have the exact exam here, but I will describe it:

- **Questions 1-10:** These were Big-O questions that related to STL data structures, priority queues, disjoint sets, and enumeration. Each question was multiple choice and had three points. Examples: "What is the running time complexity of enumerating all words of length n from an alphabet with m characters?", and "What is the running time complexity of creating a multiset from a vector with n elements?"
- **Question 11** was a bit arithmetic question, for *three points*. Example: "Let's suppose that the number y represents a set whose elements are numbers from 0 to 30. Let's also suppose that x is a number from 0 to 30. Which of these returns (y intersection { x })?"
- **Question 12** was a more "mathy" big-O question for *four points*. Example: "If *m = O(log n)*, then what is the simplest way to express *O(m+n)*?"
- **Question 13** was a *3-point* question about a priority queue or map, worth 3 points. Example: "If a node in a priority queue/heap is at index i, how do we calculate its right child?"
- **Question 14** was the same for everyone: "Explain in your own words what an interface is, and why it is a useful concept in a language like C++ (Even though C++ doesn't formally have an interface, you can view it as a restricted subset of inheritance)." *10 points*.
- **Question 15** was a priority queue push question, where you were given a picture of a priority queue, and you had to tell me the index of the value that was pushed. *4 points*. For example:



- **Question 16** was a priority queue push question, where you were given a vector representation of a priority queue, and you had to tell me the index of the value that was pushed. *4 points*. For example:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	3	21	6	25	27	38	43	54	60	34	76	50	68	39	61	84	74	59	63	87	55	41	88	90	82	94	91	85	95	86	97

Suppose I call Push(10) on this heap. What will the index of 10 be in the vector after the Push().

- **Question 17** was a priority queue push question, where you were given a vector representation of a priority queue, and you had to tell me the index of the last element in the priority queue following a pop operation. *4 points*. For example:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	19	29	48	37	33	31	68	54	83	84	86	73	72	56	89	98	71	95

Suppose I call Pop() on this heap. What will the index of 95 be in the vector after the Pop().

- **Question 18** was a disjoint set Union question. *Four points*. For example:

```
The following shows the internals for a disjoint set data structure.
We are using Union by Size.

Node:    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Links:  28  6 21  6 -1 21 -1 21  5  6 23 -1 -1 17 21 24 -1 28 16 21 16 -1  4 -1 16 16 23 16 -1
Sizes:   1  1  1  1  2  2  4  1  1  1  1  1  1  1  1  1  7  2  1  1  1  7  1  3  2  1  1  1  4

Suppose I do Union(6,16).  Mark all of the things that change in the data structure.
```

- **Question 19** was a disjoint set Find question. *Four points.* For example:

```
The following shows the internals for a disjoint set data structure.
We are using Union by Size.

Node:    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Links:  20 -1  1  9 -1  4 -1  6  9 -1 23 20  1 -1  4 23 27 23  9 -1 23  6 23 -1  7  6 19  9
Sizes:   1  3  1  1  3  1  5  2  1  6  1  1  1  1  1  1  1  1  2  3  1  1  8  1  1  1  2

Tell me the output of the following:
cout << Find(10) << " " << Find(11) << " " << Find(16) << " " << Find(27) << endl;
```

- **Question 20** was a programming question. *15 points*:

Your job is to write a program that works as follows:

- It reads two words at a time from standard input. The first word should be an integer, which we will interpret as an age, and the second word should be a string, which we will interpret as a name.
- When it's done reading input, your program should print all of the names, one name per line, in the following order:
- If person A's age is less than 80, and person A's age is less than person B's age, then person A should be printed before person B.
- If person A's age is less than 80, and person A's age is the same as person B's age, and person A's name is lexicographically less than person B's name, then person A should be printed before person B.
- If person A's and B's ages both are greater than or equal to 80, and person A's name is lexicographically less than person B's name, then person A should be printed before person B.
- Don't worry about error checking the input. Just assume it's correct.

Here's an example:

```
UNIX> cat age_name_input.txt
81 Kaitlyn
09 Christopher
33 Logan
08 Alexandra
86 Dylan
88 Addison
33 Hailey
08 Ryan
08 Alexandra
70 Owen
UNIX> a.out < age_name_input.txt
Alexandra
Alexandra
Ryan
Christopher
Hailey
Logan
Owen
Addison
Dylan
Kaitlyn
UNIX>
```

- **Question 21** was a permutation question with a twist. *15 points:*

Your job is to write a program that works as follows:

- It reads integers on standard input and puts them into a vector.
- You may assume that there are no duplicate integers.
- Let x be the last integer that is entered.
- Write a recursive program in C++ that prints all permutations of subsets of the integers that end with x.
- You may not use **next_permutation()** from the STL algorithms library.
- If you think that I'm not going to understand your program, write a small paragraph at the beginning that explains what you're doing. That can be useful for partial credit if I don't understand your code.
- Don't bother with include statements or "using namespace" statements.

Here are four examples. Your order does not have to match the order below, and I don't care if there is a space in front.

```
UNIX> g++ permutation_1.cpp
UNIX> echo 1000 | ./a.out
 1000
UNIX> echo 50 5 | ./a.out
 50 5
 5
UNIX> echo 4 6 8 | ./a.out
 4 6 8
 4 8
 6 4 8
 6 8
 8
UNIX>
```

```
UNIX> echo 4 3 2 1 | ./a.out
 4 3 2 1
 4 3 1
 4 2 3 1
 4 2 1
 4 1
 3 4 2 1
 3 4 1
 3 2 4 1
 3 2 1
 3 1
 2 3 4 1
 2 3 1
 2 4 3 1
 2 4 1
 2 1
 1
UNIX>
```