Midterm Exam

ⓘ This is a preview of the published version of the quiz

Started: Mar 15 at 11:47am

Quiz Instructions

Please answer all questions. Please answer in the format requested in the question. For example, if the question says "list the node number separated by spaces", then please don't answer "Nodes 1, 2, 3, 4". Instead, answer "1 2 3 4".

In the programming questions, you do not need to put "using namespace std" or any "#include" statements.

The point values are guidelines for the number of minutes that it should take you to do each question.

I will not be grading this on Canvas -- Instead, you will receive your grade via your "Grappling Hook" assignment.

Question 1 — 15 pts

Please answer each of the following. Don't calculate it out to be a number, but simply give an expression. Use the carat (^) for exponentiation, exclamation point (!) for factorial, and *C(n,k)* for n-choose-k. Also, please don't give an answer that is a giant sum of "n-choose-k". There is a better answer for that.

Part A: You force each of your employees to choose a password composed of exactly 5 lower-case letters followed by 4 single numeric digits. How many potential passwords are there?

Part B: You are running a lottery. You have a machine with 17 ping-pong balls, numbered 1 to 17. It has been programed to choose one of these balls at random every second. Each time a ball is chosen, it is put back onto the machine, so your machine always has the same 17 balls. Your lottery number is composed of the ping-pong ball numbers chosen in the last 6 seconds, ordered from most recently-chosen to least recently-chosen. How many different lottery numbers are there?

Part C: You are buying a new home. In your old home, you had 17 different TV's. Your new home only has room for 12 TV's, so you have to donate 5 of them to charity. How many different ways are there for you to donate those 5 TV's to charity?

Part D: There is a skyscraper in New York City whose spire has 11 lights arranged vertically. You can put a gel over a light, and that makes it a specific color. You have 11 different-colored gels. Each night, you'll use all of the gels, but you want to make sure that the gels form a different arrangement of colors every night. How many nights can you go without ever repeating an arrangement?

Part E: I'm having breakfast at a bed-and-breakfast, and my host asks me what kind of juice I want. Instead of just saying "Orange", I was dumb enough to ask what she had. She said, "I have 12 kinds of juice. You can any of them on their own, or you can combine any of them that you want in equal quantities. Or you can skip juice altogether! See, there are infinite combinations!!" You're too polite to tell her that she's wrong. How many combinations are there? (I hate to say that this is a true story, but it is):

Question 2

15 pts

3c76500

Part A: The following is an instance of disjoint sets:

```
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 2
4
Links:   1 -1 -1 10 21 -1  5 -1 18 24 24  7  5  7  1  1 24  0  5  6 24  7  0  1 -
1
```

Please enter the numbers of all nodes in the same set as node 5. Please just enter numbers separated by spaces.

Part B: The following is an instance of disjoint sets:

```
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 2
4
Links:  -1 20 20  6 21  6 -1 15 -1 23  0  6 15  5  1 21  6  5  0 16 -1  6  6 -1
  1
```

- Please enter the return value of Find(7):
- Please enter the return value of Find(18):
- Please enter the return value of Find(28):
- Please enter the return value of Find(14):

Part C: The following is an instance of disjoint sets:

```
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 2
4
Links:  17  5  8 22 19 17 16 18 -1 19  8 13 17 -1 17  8  8  8 -1  8 19 18 -1  8 -
1
```

Suppose we are using Union-By-Rank-With-Path-Compression. When we call Find(1), which node(s) change their Links field (just put node numbers separated by spaces)?

For any of the link fields in the previous question, what is the new value (just answer one)?

Question 3 — 1.5 pts

In the following procedure, suppose that the map's size is *n* and that the vector's size is *m*. What is the running time of the following procedure:

```
void p_b8ca2(const map <int, int> &x, vector <int> &y)
{
  map <int, int>::iterator xit;
```

```
  for (xit = x.begin(); xit != x.end(); xit++) {
    y.push_back(xit->first);
    y.push_back(xit->second);
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 4

1.5 pts

In the following procedure, suppose that the map's size is *n* and the vector's size is *m*. What is the running time of the following procedure:

```
int p_07659(map <int, int> &x, vector <int> &v)
{
  int nfound, index;

  index = v.size()-1;
  nfound = 0;
  do = {
    if (x.find(v[index]) != x.end()) nfound++;
    index /= 2;
  } while (index != 0);
  return nfound;
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 5

1.5 pts

In the following procedure, suppose that the set's size is *n* and that its largest element is *m*. What is the running time of the following procedure:

```
unsigned long long p_77587(const set <int> &s1)
{
  set <int>::iterator sit;
```

```
  unsigned long long s2;

  s2 = 0;
  for (sit = s1.begin(); sit != s1.end(); sit++) {
    s2 |= (1 << (*sit));
  }
  return s2;
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 6

1.5 pts

In the following procedure, suppose that the set's size is *n* and that the vector's size is *m*. What is the running time of the following procedure:

```
int p_d9450(const set <int> &x, const vector <int> &y)
{
  int n;
  size_t i;
  set <int>::iterator xit;

  n = 0;
  for (i = 0; i < y.size(); i++) {
    xit = x.lower_bound(y[i]);
    if (xit != x.end()) n += (*xit - y[i]);
  }
  return n;
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 7

1.5 pts

In the following procedure, suppose that the disjoint set has *n* elements and the vector has *m* elements. The implementation of disjoint sets is Union-By-Rank-With-Path-Compression. What is the running time of the following procedure:

```
void p_1ea1f(Disjoint_Set &x, vector <int> &set_ids)
{
  size_t i;

  for (i = 0; i < set_ids.size(); i++) {
    set_ids[i] = x.Find(set_ids[i]);
  }
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 8

1.5 pts

In the following procedure, suppose that the map's size is *n* and that its largest element is *m*. What is the running time of the following procedure:

```
void p_c3b36(map <int, int> &x)
{
  int n;

  n = x.size();
  while (x.size() > 3*n/4) x.erase(x.begin());
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 9

1.5 pts

In the following procedure, suppose that the disjoint set has *n* elements and the vector has *m* elements. The implementation of disjoint sets is Union-By-Rank-With-Path-Compression. What is the running time of the following procedure:

```
void p_014aa(Disjoint_Set &x, const vector <int> &set_ids)
{
  size_t i;

  for (i = 0; i < set_ids.size(); i += 2) {
    x.Union(set_ids[i], set_ids[i+1]);
  }
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 10

1.5 pts

In the following procedure, suppose that s1's size is *n* and that s2's size is *m*. What is the running time of the following procedure:

```
size_t p_c3b36(const string &s1, const string &s2)
{
  size_t i, found;

  found = 0;
  for (i = 0; i < s1.size(); i++) {
    if (s2.find(s1[i]) != string::npos) found++;
  }
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 11 — 1.5 pts

In the following procedure, suppose that the vector's size is *n* and that the multiset starts empty. What is the running time of the following procedure:

```
void p_5a7e8(const vector <int> &v, multiset <int> &s)
{
  size_t i;

  for (i = 0; i < v.size(); i++) s.insert(v[i]);
}
```

Your answer should be a big-O function of *n*.

Question 12 — 1.5 pts

In the following procedure, suppose that v's size is *n* and that rv's size is *m*. What is the running time of the following procedure:

```
void p_37089(const vector <int> &v, vector <int> &rv)
{
  int i, j;

  for (i = 0; i < v.size(); i++) {
    for (j = i+1; j < v.size(); j++) {
      rv.push_back(v[i] * v[j]);
    }
  }
}
```

Your answer should be a big-O function of *n* and/or *m*.

Question 13 — 8 pts

Your goal here is to write the recursive procedure binstrings(), which has the following prototype:

```
void binstrings(int n, int index, string &sofar);
```

It will be called in the main() below. It should print all *n*-letter strings composed of the characters 'A' and 'B'. You must use recursion to perform this task. You are not allowed to have a for, do or while loop in this program.

```
int main()
{
  int n;
  string s;

  cin >> n;
  s.resize('-', n);
  binstrings(n, 0, s);
  return 0;
}
```

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words

Question 14

8 pts

Your goal here is to write the procedure enum_cd(), which has the following prototype:

```
void enum_cd(int n);
```

It will be called in the main() below. It should print all *n*-letter strings composed of the characters 'C' and 'D'. You should use one of the enumerations that you learned in this class to perform this task.

```
int main()
{
  int n;

  cin >> n;
  enum_cd(n);
  return 0;
}
```

Edit View Insert Format Tools Table

12pt Paragraph

p 0 words

Question 15 — 14 pts

Behold the header file **collection.hpp**:

```
#pragma once
#include <string>
using namespace std;

class Collection {
  public:
    virtual ~Collection() {};
    virtual void Add_Item(int id, const string &thing) = 0;
    virtual void Print() = 0;
};
```

A collection stores items. Each item has an integer id, and then some collection of "things". Each thing is defined by a string, and you can have multiple things with the same string. The methods are defined as follows:

- **Add_Item()** adds the item with the given id to the collection, if it's not there already. It then adds the string "thing" to the item's collection of strings.
- **Print()** prints out the items, one per line. To print an item, it should print the item's id followed by a colon (no space between the id and the colon). It should then print the "things" that belong to the item, each separated by a space. It doesn't matter what order you print out the items or the things.

You are going to implement the class **MyImp**, which will implement the **Collection** interface. Do so below, first by specifying the **MyImp** class as you would in a header file (**myimp.hpp**), and then by implementing the methods as you would in an implementation file (**myimp.cpp**).

Here's an example **main()**;

```
#include <iostream>
#include "collection.hpp"
#include "myimp.hpp"
using namespace std;

int main()
{
  int id;
  string item;
  Collection *c;

  c = new MyImp;

  while (cin >> id >> item) c->Add_Item(id, item);
  c->Print();
  return 0;
}
```

And here's an example of it compiled to **a.out** and running:

```
UNIX> cat items.txt
23894 dog
23894 cat
854 cat
23894 cat
7 fred
7 binky
UNIX> a.out < items.txt
7: fred binky
854: cat
23894: dog cat cat
UNIX>
```

I'm giving you flexibility in how you implement this, so although that's a legal output above, there are many other legal outputs. For example the following outputs are also legal:

```
UNIX> a.out < items.txt
23894: dog cat cat
7: fred binky
854: cat
UNIX>
```

or

```
UNIX> a.out < items.txt
854: cat
23894: cat dog cat
7: binky fred
UNIX>
```

You need to implement this so that it follows the specification and is efficient. Obviously, you'll have to use data structures from the STL.

Edit View Insert Format Tools Table

12pt Paragraph

p

0 words

Saving...

Submit Quiz