CS302 Midterm Exam - October 11, 2016

Do your answers on the answer sheets provided. When you write code, you do not need to have "include" or "using" statements.

Question 1

You are using an implementation of disjoint sets, and when you call the **Print()** method, it prints the following:

```
Elts:   0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
Links:  7 -1 -1  7  2 -1  0  1  1  2 -1  1  1  5  9  1
Ranks:  2  9  4  1  1  2  1  4  1  2  1  1  1  1  1  1
```

Part A: On the answer sheet, please list all of the disjoint sets. To be clear on what I want, it is all of the sets that are represented by the preceding printout. An example answer (that is incorrect, but its format is correct) would be the following.

> "{0, 1, 2, 3}, {4, 5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15}"

Part B: Which implementation was used to generate that output? Union-by-size, Union-by-height, or Union-by-rank-with-path-compression?

Part C: Please tell me the return values of the following calls, if they are made in the following order:

- `d.Find(0);`
- `d.Find(15);`
- `d.Find(14);`
- `d.Find(10);`
- `d.Union(1, 2);`

Part D: Suppose you call **new PQueue(v)**, where **v** is the following vector of doubles:

```
{ 75, 56, 86, 40, 46, 71, 68, 10, 32, 16, 20, 5 }
```

On the answer sheet, please give me the vector **h** that is the protected member variable of the **PQueue** class, after the **new** call. You should use the technique that is *O(n)*, where *n* is the size of the vector.

Question 2

On the answer sheet, please tell me the running times of the following procedures. In every procedure, `C.size()` is n. I want your answers to be big-O of functions of n, such as $O(n)$ or $O(n^2)$. I don't want answers like $O(c.size())$, or "quadratic."

```
void Proc_A(list <double> &C)
{
  list <double>::iterator a;
  list <double>::iterator b;

  b = C.begin();
  for (a = C.begin(); a != C.end(); a++) {
    if (a != C.begin()) {
      C.insert(b, *a);
      b++;
    }
  }
}
```

```
void Proc_B(vector <double> &C)
{
  int i, j;

  for (i = 0; i < C.size(); i++) {
    for (j = 0; j <= i; j++) {
      C[i] += C[j];
    }
  }
}
```

```
vector <double> Proc_C(vector <double> C)
{
  vector <double> rv;
  int i, j;
  double x;

  for (i = 0; i < (1 << C.size()); i++) {
    x = 0;
    for (j = 0; j < C.size(); j++) {
      if (i & (1 << j)) x += C[j];
    }
    rv.push_back(x);
  }
  return rv;
}
```

```
vector <double> Proc_D(set <double> C)
{
  set <double>::iterator cit;
  vector <double> rv;

  for (cit = C.begin(); cit != C.end(); cit++) {
    rv.push_back(*cit);
  }
  return rv;
}
```

```
PQueue *Proc_E(vector <double> &C)
{
  return new PQueue(C);
}
```

```
vector <double> Proc_F(PQueue &C)
{
  vector <double> rv;

  while (C.size() > 0) rv.push_back(C.Pop());
  return rv;
}
```

```
void Proc_G(multimap <double, int> &C)
{
  int i, n;

  n = C.size();
  for (i = 0; i < n; i++) {
     C.insert(make_pair(drand48(), i));
  }
}
```

```
multimap <double, int> Proc_H(vector <double> C)
{
  int i;
  multimap <double, int> rv;

  for (i = 0; i < C.size(); i++) {
    rv.insert(make_pair(C[i], i));
  }
  return rv;
}
```

```
void Proc_I(int n)
{
  Disjoint J(n);
  int i;

  for (i = 0; i < n; i += 2) J.Union(i, i+1);
}
```

```
vector <int> proc(Disjoint &C, int n)
{
  vector <int> rv;
  int i;

  for (i = 0; i < n; i++) rv.push_back(C.Find(i));
  return rv;
}
```

Question 3

Part A: Write the procedure **proc()** with the following prototype:

```
void proc(int n, string s);
```

```
aa
ba
ab
bb
```

What **proc()** should do is print out every **n**-character string composed of the letters in **s**, where repeating letters is ok. For example, if **n** is 2 and **s** is "ab", then your program should print the strings to the right.

You may assume that there are no repeated characters in **s** and that the total number of strings being printed will be less than 2^{31}. You can also print the strings in any order that you want -- just no repeated or extra strings.

Part B: Exactly how many strings should **proc()** print, as a function of **n** and **s.size()**?

Question 4

You are to write the procedure **do_transpose()** that has the following prototype:

```
vector <int> do_transpose(vector <int> m);
```

The parameter **m** contains a square matrix of bits. The number of rows and columns is the size of **m**, and that is guaranteed to be 32 or smaller. Bit *i* of **m[j]** contains the bit in row *j* column *i*. So, for example, if **m** is: `{ 0x3, 0xa, 0xf, 0x7 },` then that corresponds to the matrix

```
1 1 0 0
0 1 0 1
1 1 1 1
1 1 1 0
```

Your procedure should return the transpose of **m**, in the same format. The formal definition of a transpose is as follows: If **T** is the transpose of **M**, then **T[i,j]** is equal to **M[j,i]**.

For example, the transpose of the matrix above is:

```
1 0 1 1
1 1 1 1
0 0 1 1
0 1 1 0
```

Which is equal to: `{ 0xd, 0xf, 0xc, 0x6 }.`

Question 5

Write a **main()** program for the procedure in Question 4. It should do the following:

- It should treat each word on the command line as a number in hexadecimal that starts with "0x".
- It should read each of these words and create a vector of ints called **m**.
- It should call **do_transpose()** on **m**, and print the return vector in hexadecimal (again, with the leading "0x"), one per line.
- It should error check the input. If there is an input error, then it should print an error message and exit immediately.

So, for example:

```
UNIX> a.out 0x3 0xa 0xf 0x7
0xd
0xf
0xc
0x6
UNIX> a.out 3 a f 7
Bad command line argument 3
UNIX> a.out 0xf 0xf
Number too big for a matrix with 2 rows: 0xf
UNIX>
```