Final Exam

- Time: July 27, 4:30pm
- Accumulative. Covers all materials.
  - roughly 70% for after midterm materials.
- Some questions may be derived from projects, but you don’t have to memorize project details.
- You can use two piece of letter-size cheating sheets, created by hand.
  - Machine printouts will be confiscated.

- Except a calculator no other equipment can be used.
  - no notebook, cell phone, beeper, magnifier, etc..
  - you lose 10 points each time your cell phone rings
- If you need special arrangements for the final or course grade, present your requests before the final.
  - I will not make any deal after the final, period.
Course Grade

- We will have an extra office hour after the final, but before I submit your grade to Univ., for you to check your final exam and address any concern you may have.
- Before the last office hours, you will be informed via email of your final exam points, your record of the entire semester, the equations used to calculate your course grade, and your course grade.
  - The TA will also be available during the session.
  - This will be your last chance to set the record straight.
  - It will be very difficult to change your grade after I submit it, and I will be very reluctant to do so.
  - The time and date will be announced before the final; please watch the course web page.

Disclaimers

- This review is provided as a helper; it is by no means comprehensive.
- Study all course materials thoroughly.
Inheritance and Polymorphism

You are expected to

- understand the basic concepts;
- be familiar with related terminologies, such as base/derived/abstract classes, virtual functions, protected data or methods, and so forth.
- read a code segment that uses inheritance and/or polymorphism and predict execution results;

1. Consider the C++ code below.

```cpp
class X {
    protected:
    int a;
    public:
    X () {a = 1;}
    void f() {a += 10;}
    virtual void g() {a += 100;}
};

class Y : public X {
    public:
    Y () : X() {}  
    void f() {a -= 10;}
};

class Z : public X {
    public:
    Z() : X() {}  
    void g() {a -= 100;}
```
main()
{
    X x, *p;
    Y y, *q;
    Z z, *r;
    p=&x;  p->f();  p->g();
    q=&y;  q->f();  q->g();
    q=&z;  q->f();  q->g();
    r=&z;  r->f();  r->g();
}

What are the values of x.a, y.a, and z.a after the completion of the last statement?

---

Hash Tables

You are expected to

- be familiar with concepts and terminologies
- perform insertion operations with hash tables using chaining, linear probing, and quadratic probing.
- given load factor $\lambda$, compute average number of probings
- able to create small segment of codes
Consider a hash table with CAPACITY=11 that uses the division hash function and quadratic probing. Start with the table empty, and insert 1, 12, 17, 23, 34, and 45 to the table, in that order. Show the contents of the table.

A hash table $A$ is used by the university to store the records of students. The table does not provide a remove() method. Instead, each student record contains a status field with value 1 indicating the student is currently enrolled, and 2 indicating that the student has graduated. At the end of each academic year, all the entries in table $A$ are retrieved and inserted to a new hash table $B$, except those who have graduated for $K$ or more years. Subsequently, table $A$ is deleted, and table $B$ is named table $A$. We further make the following assumptions:

- Hash tables in this question have CAPACITY=20000 and use linear probing to resolve conflicts.
- The university has an average of 5000 students. That is to say, when the hash table $A$ is “re-born,” it is expected to contain 5000 records.

- The university admits 1000 freshmen per year, and every year 1000 students graduate.

Calculate the largest $K$ value such that the average number of table entries examined for each search operation is equal to or less than 2.

### AVL Trees

You are expected to

- understand the definition and terminology
- be able to perform AVL insertion
- understand the four cases of insertions: LL, RR, RL, LR (including proof)
- be familiar with implementation issues
- be able to create small segments of codes
Model Question

Starting from the AVL tree below, show the result after inserting 5 and 81.

```
        45
       / \
      20  82
     /   / \
    10  77  88
     |     |     |
    69  80  85  95
```

Graphs

You need to be able to

1. understand important graph terminologies, such as, the connectivity of a graph, DAG, graph traversals, etc.;
2. draw a graph, given its vertices and relationship (that is, edges) among vertices;
3. draw a graph, given its adjacency matrix/list representation;
4. draw the adjacency matrix or adjacency list representation of a given graph;
5. perform topological sorting, Dijkstra and Prim algorithms;
1. Let $G = (V, E)$ where $V = \{\text{dark, sun, lion, phone, system, gray, oxford}\}$ and an edge $(u, v)$ is in $E$ if and only if word $u$ proceeds word $v$ in a dictionary and if $u$ and $v$ have at least one letter in common. Draw graph $G$.

2. Determine if the following statements about graphs are true or false.
   (a) A graph could contain 10000 nodes and 0 edges.
   (b) A graph could contain 0 nodes and 10000 edges.
   (c) A connected, directed graph is a graph in which any vertex is connected all the other vertices by a direct link.
   (d) An undirected graph is a connected graph.
   (e) A tree is a connected graph.
   (f) A BFS order of a DAG is also a topological order of the DAG.
   (g) A DAG is a connected, directed graph.
3. Answer the following questions based on the DAG:

(a) Give a DFS sequence that is not a topological order.
(b) Give a BFS sequence that is not a topological order.
(c) If each vertex in the DAG is a task and the task is performed when the vertex is “printed” in the topological sort algorithm. Show the contents of the $\text{pred}$ array after 3 tasks have been performed. (you also have to give which three tasks have been performed.)

4. Perform Dijkstra’s shortest path algorithm on the graph below, using vertex 3 as the starting point, and show the contents of the $\text{dist}$ and $\text{pred}$ arrays after three nodes have been marked and the distances and predecessors of their neighbors updated.
Iterators

You are expected to

- understand the concept
- be familiar with relevant C++ concepts
  - operator overloading, the difference between pre and post increment operators, etc.
- implement short code segments involving iterators

Model Question

Consider a class that maintains the academic records of students.

```cpp
class Students{
public:
    add (string name, float avg_gpa);
    update_gpa (string name, float new_gpa);
    remove (string name);
private:
    struct StuRecord {
        string name;  // including both first and last names
        float avg_gpa;
    };
    StuRecord students[1000];
    int no_of_students;
};
```
Our goal is to develop an outstanding-students-iterator, called OSI, which enumerates all students with GPA above a given threshold. For example, the following loop will print the names of all GPA-3.75-and-above students in a Students object stu.

```cpp
OSI p;
for (p=stu.first_os(3.75); p!=stu.end_os(); ++p)
    cout << *p << "\n";
```

Answer the following questions.

1. Which pieces of information need to be stored in an OSI?

2. Implement `first_os()` method of class Students.
3. Implement the deference operator of class OSI.

4. Implement the pre-increment operator of class OSI.
Red-Black Trees

You are expected to

- understand the concept
- be familiar with definitions
- Draw a red-black tree from a B-tree, and vice versa.
- Perform insertions and deletions (they are really B-tree operations)

Model Questions

Draw the 2-3 tree below as a red-black tree.

```
   30, 40
  /     \
10, 20  37  50
 /   /   /   /
5  15,18 25  35 38 45 55
```