Print Your Name:

Read the following now.

- You have 100 minutes to work on 9 questions, totaling 200 points.
- To ensure partial credit, show all your work!
- Write your name on all pages of the exam.
- Write down your answers clearly. I reserve the right to take off points due to poor writing or English structures.
- One blank page is provided at the end for your convenience.

STOP! Do not turn to the next page until instructed to do so.
1. (20 points in total) Consider the array representation of a maximum heap:

   85, 43, 66, 10, 40, 65, 20, 6, 5, 36, 35, 34

(a) (10pt) Show the heap as a tree after inserting 99.

(b) (10pt) Starting from the original heap, show the heap as an array after removing the top element, 85.
2. (25 points in total) Given the AVL tree below, answer the following questions.

(a) (5pt) Name the sibling of node 55.

(b) (5pt) Consider the insertion of 57 to the tree, name the A node, that is, the first ancestor of 80 who loses balance due to the addition of 80.

(c) (10pt) Show the tree after the insertion of 57 is completed.

(d) (5pt) Consider the original tree (the one shown above, without 80 in it). The tree is not a minimum AVL tree of height 4. Name a node whose removal will make tree the minimal.
3. (35 points in total) Consider the C++ classes below.

```cpp
class X {
    int a, b;
    X() {a=10; b=256;}
    virtual void f () = 0;
    void g () {a *= 2;}
};

class Y : public X {
    Y() : X() {} // the a and b of Y are initialized
        // by the constructor of X
    f () {b /= 2;}
};

class Z : public Y {
    Z() : Y() {} // the a and b of Z are initialized
        // by the constructor of Y, hence by
        // the constructor of X
    f () {b /= 4;}
    g () {a *= 4;}
};

class W : public X {
    int c;
    W() : X() {c=27} // the a and b of W are initialized
        // by the constructor of X
    f () {b /= 8;}
    h () {c /= 3;}
};

Y y;
Z z;
W w;
X *x_ptr = &z;
```

(a) (5pt) Name the base class(es) in the above code.
(b) (5pt) Explain *abstract class* in two sentences or less.

(c) (2pt) Name the abstract class(es) in the above code.

(d) (6pt) Give the values of `y.a` and `y.b` after the execution of `y.f()` and `y.g()`.

(e) (6pt) Give the values of `z.a` and `z.b` after the execution of `z.f()` and `z.g()`.

(f) (6pt) Let us reinitialize `z`, that is, `z.a=10` and `z.b=256`. Give the values of `z.a` and `z.b` after the execution of `x_ptr->f()` and `x_ptr->g()`.

(g) (5pt) Circle the C++ statements below that are illegal.

- `w = y;`
- `z = y;`
- `y = z;`
- `x_ptr = new W;`
- `x_ptr = new X;`
4. (15pt) Remove 66 from the B-tree below (MINIMUM=2), and show the result.

```
   44, 77
   /     \
 10,20,30 60,70 80,90
 /     /     /  \
1,2  15,17  24,25,26 32,35  55,56  65,66  75,76  78,79  81,82  91,92
```

5. (15pt) Consider a hash table with CAPACITY=13 that uses the division hash function and quadratic probing. Start with the table empty. Insert 19, 15, 16, 136, 24, and 41, to the table, in that order. Show the contents of the table.

```
0 1 2 3 4 5 6 7 8 9 10 11 12
- - - - - - - - - - - - -
```
6. (25 points in total) Consider a class that maintains the academic records of students.

```cpp
struct StuRecord {
    string name; // including both first and last names
    float avg_gpa;
};
struct ListNode {
    StuRecord stu_rec;
    ListNode* next;
};
class Students{
public:
    add (string name, float avg_gpa);
    update_gpa (string name, float new_gpa);
    remove (string name);
private:
    ListNode* head; // a pointer to the first ListNode
};
```

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

(a) (8pt) Declare the data members of class OSI. Don’t worry about constructors, destructors, and any other method.

```cpp
class OSI {
private: // declare your data members here
public: // constructors/methods here; not required in your answer
};
```
(b) (7pt) Implement the dereference operator of class OSI.

    StuRecord& OSI::operator* ()
    {
    }

(c) (10pt) Implement the pre-increment operator of class OSI.

    OSI OSI::operator++ ()
    {
    }
7. (25 points in total) Consider the adjacency matrix representation of graph $G$:

$$
\begin{array}{c|cccccc}
  & A & B & C & D & E & F \\
  A & 0 & 1 & 1 & 0 & 0 & 1 \\
  B & 0 & 0 & 0 & 0 & 0 & 0 \\
  C & 1 & 1 & 0 & 1 & 1 & 0 \\
  D & 0 & 0 & 0 & 0 & 0 & 1 \\
  E & 0 & 0 & 1 & 0 & 0 & 1 \\
  F & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
$$

(a) (10pt) Draw graph $G$.

```
F     A
     
D     B
     
E     C
```

(b) (5pt) Explain why $G$ is not connected.

(c) (10pt) Show the adjacency list representation of $G$.

```
A
B
C
D
E
F
```
8. (20 points in total) Perform Dijkstra’s shortest path algorithm on the directed graph shown below, using 7 as the starting point.

(a) (5pt) Show the initial contents of the dist and pred arrays.

(b) (15pt) Show the contents of the dist and pred arrays after the completion of 4 iterations.
9. (20 points in total) Perform Kruskal’s minimum spanning tree algorithm on the graph shown below. Consider the first four edges that are admitted to the spanning tree.

- (5pt) Identify the first edge, and show the trees as defined by the parent relationship of vertices after the selection of the first edge.

- (5pt) Identify the second edge, and show the trees as defined by the parent relationship of vertices after the selection of the first edge.

- (5pt) Identify the second edge, and show the trees as defined by the parent relationship of vertices after the selection of the first edge.

- (5pt) Identify the forth edge, and show the trees as defined by the parent relationship of vertices after the selection of the first edge.
*This page is intentionally left blank.*