Paper I

COMP 2012 Final Exam - Fall 2018 - HKUST

Date: December 11, 2018 (Tuesday)
Time Allowed: 3 hours, 8:30am–11:30am
Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **7** questions on **34** pages (including this cover page, appendix and 4 blank pages at the end) printed on **2** sets of papers:
   - Paper I: Description of problem 1 - 5 **AND** space for **ALL** your answers.
   - Paper II: Description of problem 6 - 7.
3. Write your answers in the space provided in paper I.
4. All programming codes in your answers must be written in the ANSI C++ version as taught in the class.
5. For programming questions, unless otherwise stated, you are **NOT** allowed to define additional structures, classes, helper functions and use global variables, nor any library functions not mentioned in the questions.

Student Name	
Student ID	
Email Address	
Venue	LG1 Table Tennis Room / Tsang Shiu Tim Art Hall * (* Delete as appropriate)
Seat Number	

For T.A. Use Only

Problem	Topic	Score
1	True or False Questions	/ 10
2	Standard Template Library (STL)	/ 6
3	Order of Construction and Destruction	/ 11
4	AVL Tree	/ 7
5	Hashing	/ 12
6	Inheritance, Polymorphism & Dynamic Binding	/ 28
7	Binary Search Tree (BST)	/ 26
Total		/ 100

Problem 1 [10 points] True or False Questions

Indicate whether the following statements are *true* or *false* by circling **T** or **F**.

T **F** (a) Subscript operator, `operator[]`, can be overloaded to accept multiple arguments.

T **F** (b) The following is an invalid template declaration, that is it CANNOT be instantiated into a function.

```
template <int x>
int function() {
  return x;
}
```

T **F** (c) There is NO compilation error in the following program.

```
#include <iostream>
using namespace std;

template <typename T>
void f(T x, T y) {
  cout << "Template" << endl;
}

void f(int x, int y) {
  cout << "Non-template" << endl;
}

int main() {
  f(1, 2);
  f('a', 'b');
  f(1, 'b');
}
```

T **F** (d) A `const_iterator` is an iterator that cannot be used to modify the element to which it refers to.

T **F** (e) A privately inherited class can be inherited further.

T **F** (f) There is NO compilation error in the following program.

```
class Base { };

class Derived : private Base { };

int main() {
  Derived d;
  Base& bRef = d;
}
```

T **F** (g) Dynamic binding is done for the function call at line 16 of the following program.

```
1  #include <iostream>
2  using namespace std;
3
4  class A {
5    public:
6      virtual void f() { cout << "A::f()" << endl; }
7  };
8
9  class B : public A {
10   public:
11     virtual void f() { cout << "B::f()" << endl; }
12 };
13
14 int main() {
15   A* aPtr = new B;
16   aPtr->A::f();
17 }
```

T **F** (h) A static data member may be initialized while defining it.

T **F** (i) Declaring a friend function in private section of a class rather than public alters its meaning.

T **F** (j) It is sufficient to construct the original Binary Search Tree from the preorder traversal sequence.

Problem 2 [6 points] Standard Template Library (STL)

Define the `print` function which prints the integers contained in a STL vector container. The order of the integers printed should be the same as they are stored in the container, and they should be separated by a single comma. For example, with the following program:

```
#include <iostream>
#include <vector>
using namespace std;

// Your complete print function definition will be put here

int main()
{
    vector<int> v;
    v.push_back(3);
    v.push_back(2);
    v.push_back(4);
    v.push_back(5);
    print(v);

    return 0;
}
```

The `print` function should print

```
3,2,4,5
```

on the screen.

Note: You MUST make use of iterators in this question, and you CANNOT use `size()` and `operator[]` of the `vector` class.

Answer:

Problem 3 [11 points] Order of Construction and Destruction

```
#include <iostream>
using namespace std;

class Weapon { };

class Sword : public Weapon {
  public:
    Sword(int n = 1) { cout << n << " x S" << endl; }
    ~Sword() { cout << "~S" << endl; }
};

class Hero {
    Weapon* w;
  public:
    Hero() { cout << "H" << endl; w = new Weapon; }
    ~Hero() { cout << "~H" << endl; delete w; }
    virtual const Hero& operator=(const Hero& h) { cout << "H=" << endl; }
};

class SpiderMan : public Hero {
    Weapon* w;
  public:
    SpiderMan() { cout << "SM" << endl; w = new Weapon; }
    SpiderMan(const SpiderMan& s) { cout << "Copy SM" << endl; w = new Weapon; }
    virtual ~SpiderMan() { cout << "~SM" << endl; delete w; }
    const SpiderMan& operator=(const SpiderMan& s) { cout << "SM=" << endl; }
};

class DeadPool : public SpiderMan {
    Sword sword;
  public:
    DeadPool() : sword(2) { cout << "DP" << endl; }
    DeadPool(const DeadPool& d) : SpiderMan(d) { cout << "Copy DP" << endl; }
    ~DeadPool() { cout << "~DP" << endl; }
    const DeadPool& operator=(const DeadPool& d) { cout << "DP=" << endl; }
};

int main() {
  cout << "--- Block 1 ---" << endl;
  Hero* hero = new DeadPool;
  cout << "--- Block 2 ---" << endl;
  SpiderMan* spiderman = new DeadPool;
  cout << "--- Block 3 ---" << endl;
  DeadPool deadpool(*dynamic_cast<DeadPool*>(spiderman));
  cout << "--- Block 4 ---" << endl;
  *hero = *spiderman;
  cout << "--- Block 5 ---" << endl;
  delete hero;
  cout << "--- Block 6 ---" << endl;
  delete spiderman;
}
```

Write down the output of the above program when it is run. Some lines of outputs are already given. Assume the compiler DOES NOT do any optimization.

Answer:

```
--- Block 1 ---
```

```
--- Block 2 ---
```

```
--- Block 3 ---
```

```
--- Block 4 ---
```

```
--- Block 5 ---
```

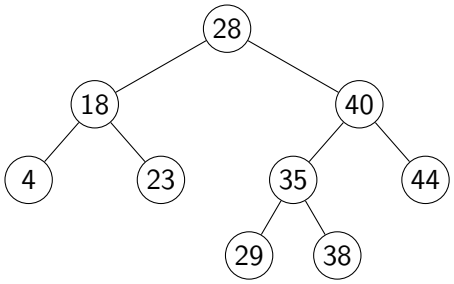
```
--- Block 6 ---
```

Problem 4 [7 points] AVL Tree

(a) [1 point] What is the maximum height of any AVL tree with 7 nodes? Assume that the height of a tree with a single node is 0.

Answer:

(b) [6 points] Consider the following AVL tree.



Insert the key 36 into the tree and re-balance if needed. Draw all the intermediate trees (including the tree after insertion and each rotation, if any) and final tree. You must use the algorithms discussed in class for inserting and re-balancing.

Answer:

Problem 5 [12 points] Hashing

Fill in the following hash tables to show their contents when different open addressing methods are used to insert the following 6 strings (in the given order) into the tables of size 7: "best", "stapler", "learn", "bets", "plaster", "renal", and compute the total number of probes for hashing the six strings for each method.

Assume the mapping of English alphabets to numeric codes is as follows:

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m
Code	1	2	3	4	5	6	7	8	9	10	11	12	13
Alphabet	n	o	p	q	r	s	t	u	v	w	x	y	z
Code	14	15	16	17	18	19	20	21	22	23	24	25	26

and the following hash function is to be used for this question:

$$hash(s) = (\text{code of } s[0] + \text{code of } s[1] + \ldots + \text{code of } s[L-1]) \bmod 7$$

where s is the input string (key) to be hashed and L is the key length.

For example, the hash value of string key "best" is computed as follows.
'b' = 2, 'e' = 5, 's' = 19, 't' = 20
hash value = (2 + 5 + 19 + 20) mod 7 = 46 mod 7 = 4

Note that, once a key value gets into the hash table, it stays in the table. To illustrate that, the insertion of the first key "best" has been done for you in the tables below.

(a) [4 points] Linear probing: $h_i(s) = (hash(s) + i) \bmod 7$

Table Index	**Insert best**	**Insert stapler**	**Insert learn**	**Insert bets**	**Insert plaster**	**Insert renal**
0						
1						
2						
3						
4	best	best	best	best	best	best
5						
6						

The total number of probes for hashing the six strings (including hashing "best"):

(b) [4 points] Quadratic probing: $h_i(s) = (hash(s) + i^2) \bmod 7$

Table Index	Insert best	Insert stapler	Insert learn	Insert bets	Insert plaster	Insert renal
0						
1						
2						
3						
4	best	best	best	best	best	best
5						
6						

The total number of probes for hashing the six strings (including hashing "best"):

(c) [4 points] Double hashing: Use the $hash'(k)$ function below as the second hash function:

$$hash'(s) = 5 - (\text{code of } s[0] + \text{code of } s[1] + \ldots + \text{code of } s[L-1]) \bmod 5$$

Table Index	Insert best	Insert stapler	Insert learn	Insert bets	Insert plaster	Insert renal
0						
1						
2						
3						
4	best	best	best	best	best	best
5						
6						

The total number of probes for hashing the six strings (including hashing "best"):

Problem 6 [28 points] Inheritance, Polymorphism and Dynamic Binding

Please refer to Paper II for the problem description.

Based on the given information, complete the implementation of '`PocketMonster`' class, '`ElectricPocketMonster`' class and '`Team`' class in their respective .cpp files, namely, "`PocketMonster.cpp`", "`ElectricPocketMonster.cpp`" and "`Team.cpp`" respectively.

(a) [6 points] Implement the member function

```
virtual void battle(PocketMonster& enemy);
```

of the class '`PocketMonster`' in a separate file called "`PocketMonster.cpp`". Include all the required header file(s) to make sure the program compiles.

Answer: `/* File: PocketMonster.cpp */`

(b) [11 points] Implement the following 3 member functions

- `ElectricPocketMonster(string name, int hp, int attack, int defense, int speed, int exp, string bEvolve, string aEvolve, int electricPower);`
- `void battle(PocketMonster& enemy);`
- `void print() const;`

of the class '`ElectricPocketMonster`' in a separate file called "`ElectricPocketMonster.cpp`". Include all the required header file(s) to make sure the program compiles.

Answer: `/* File: ElectricPocketMonster.cpp */`

(c) [11 points] Implement the following 4 member functions

- `Team(const Team& t);`
- `~Team();`
- `void addMember(PocketMonster* pm);`
- `void print() const;`

of the class '`Team`' in a separate file called "`Team.cpp`". Include all the required header file(s) to make sure the program compiles.

Answer: `/* File: Team.cpp */`

Problem 7 [26 points] Binary Search Tree (BST)

Please refer to Paper II for the problem description.

Based on the given information, complete "`Solution.tpp`" in the space below. Implement your functions in the corresponding places.

(a) [5 points] Implement

```
bool insert(const T& x, BinaryNode*& t);
```

below.

Answer:

(b) [6 points] Implement

```
bool remove(const T &x, BinaryNode*& t);
```

below.

Answer:

(c) [4 points] Implement

```
void printMax() const;
```

below.

Answer:

(d) [4 points] Implement

```
bool isFull(BinaryNode* t) const;
```

below.

Answer:

(e) [3 points] Implement

```
BinaryNode* clone(BinaryNode* t) const;
```

below.

Answer:

(f) [4 points] Implement

```
void makeEmpty(BinaryNode* t);
```

below.

Answer:

---------- END OF PAPER ----------