上 海 交 通 大 学　　卷

(2019−2020 Academic Year/Fall Semester)

Class No. _____ Student ID No. _____

Name in English/Pinyin: _____ Name in Hanzi, if applicable: _____

VE280 Programming and Elementary Data Structures

Final Exam

August 4, 2020, 10:00am-11:40am

The exam paper has **10** pages in total.

Exam rules and information:

- The five main sections of this exam are independent.
- This is an open-book and open-note exam.
- No communication devices/apps are allowed. These include cell phones and smart watches.
- Read very carefully the instructions in each question.
- For coding questions, you are responsible to make sure that your code works on Linux. You can compile with `g++ -c` to ensure that there's no syntactic errors.

Instructions for submission:

- Work in the directory where you've found this pdf file. This will be your working directory. Answer the questions in the corresponding files. The files `Q1.1.txt` to `Q1.10.txt` are for the first problem, the files `Q2.1.txt` to `Q2.6.txt` are for the second problem, the file `Q3.txt` is for the third problem, the file `dict.hpp` is for the fourth problem, and `binaryTree.cpp` is for the last problem.
- When the time is up, you will prepare a tar file of this directory. If you are in the parent directory, you can use the following command, where `XYZ` is your directory name:

```
tar -cf XYZ.tar XYZ
```

- Submit the tar file on Canvas.

You are to abide by the University of Michigan-Shanghai Jiao Tong University Joint Institute (UM-SJTU JI) honor code. Please sign below to signify that you have kept the honor code pledge.

THE UM-SJTU JI HONOR CODE

I accept the letter and spirit of the honor code:
I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code by myself or others.

Signature:
Log in on Canvas to sign via the corresponding Quiz.

1 General Programming Related Questions (20 pts)

Provide your answers in the corresponding files `Q1.1.txt` to `Q1.10.txt`.

1. (2 points) Consider the following templated class `Set` based on the STL container `vector`:

```
template <class Key>
class Set{
   vector<Key> keys;
public:
   Set();
   bool find(Key k);
   void insert(Key k);
   bool remove(Key k);
   int size();
};
```

   Why can we say that it is not a pure implementation of the Set ADT from the point of view of the user of the ADT?

2. (2 points) Which C++ construct could we use to obtain a purer implementation of the ADT in the previous question? You don't need to provide any code.

3. (2 points) Consider the following classes where `A` is the implementation of an ADT:

```
class A{
   int a;
public:
   virtual void f();
};
class B: public class A{
public:
   void f();
}
```

   Can we always say that `B` is the implementation of an ADT subtype of `A`?

4. (2 points) Given a class `Vector` which represents a vector of real values, how do you declare an overloaded substraction operator that takes two vectors and returns a vector that is given by the first vector minus the second vector, as a member function of the class? Assume the declaration is put inside the class definition.

5. (2 points) Consider the following classes where `A` is the implementation of an ADT:

```
class A{
   int i;
public:
   A(): i(0){
      cout << "dc";
   }
   A(const A &a){
      cout << "cc";
      i = a.i;
   }
   A &operator=(const A &a){
      cout << "a";
      i = a.i;
      return *this;
   }
   void f(A a){
      cout << a.i;
   }
};
int foo(){
   A a;
   a.f(a);
}
```

What is the output of executing the `foo` function?

6. (2 points) Consider the `IntSet` class that is based on a dynamic array, whose size is doubled when its full capacity is reached. Assume that we create an empty `IntSet` with a capacity of 8, then insert 65 integers. How many times in the worse case will the dynamic array grow? Provide your answer in numeric form, e.g., 10.

7. (2 points) How many number of elements are copied in total (when the array is doubled) after all the insertions in the previous question?

8. (2 points) Assume we have sorted values stored in a double-ended doubly-linked list. What is the complexity of inserting a value in that list such that the list stays sorted after the insertion? Choose between $O(1)$, $0(\log(n))$, $O(n)$, or $O(2^n)$ where n is the number of elements in the list.

9. (2 points) Suppose that `List` is a templated container over type T. However, it stores object by pointers-to-T, instead of the actual instance T. For the following code, write which of the at-most-once invariant or the rules of Existence, Ownership, and Conservation it violates, or "none" if it does not violate any of them.

```
List<int> l1 = new List<int>();
```

```
List<int> l2 = new List<int>();
for(int i=0; i<10; i++){
   int *ip = new int(i);
   l1.insert(ip);
   l2.insert(ip);
}
```

10. (2 points) Using the STL containers, how can we define a variable named `lm` that is a list of map containing pairs of string and int?

2 Correctness (24 pts)

For each of the following codes, there is **at most one** major problem. If there is one, explain what is the problem. You don't need to tell us how to fix it. If there is none, write "None" without the quotes.

Provide your answers in the corresponding files `Q2.1.txt` to `Q2.6.txt`.

1. (4 points)

```
class foo{
   int a,b;

public:
   foo(int _a = 0, int _b = 0);
   // Constructor with default arguments. It assigns _a to data
   // member a and _b to data member b. Implementation omitted.
   virtual void setVal(int _a, int _b);
   // MODIFIES: this
   // EFFECTS: set data member a to _a and data member b to _b.
   //          Implementation omitted.
};

class bar: public foo{
public:
   bar();
   // Default constructor
   int sum();
   // EFFECTS: return the sum of data member a and b.
};

int bar::sum(){
   return a+b;
```

```
}
```

2. (4 points)

```
class A{
   int a;
public:
   A(int _a=0): a(_a){ }
   int getA(){ return a; }
};

class B: public A{
   int b;
public:
   B(int _b=0): b(_b){ }
   virtual int getB() { return b; }
};

int main() {
   A *pa = new B();
   B *pb = dynamic_cast<B *>(pa);
   return pb->getB();
}
```

3. (4 points)

```
class foo{
   int a,b;

public:
   foo(int _a, int _b=0);
   // Constructor with default argument. It assigns _a to data
   // member a and _b to data member b. Implementation omitted.
   void set(int _a=0, int _b=0);
   // MODIFIES: this
   // EFFECTS: set data member a to _a and data member b to _b.
   //          Implementation omitted.
};

class bar: public foo{
```

```
public:
   bar();
   // Default constructor
   void set(int a=1, int b=1);
   // MODIFIES: this
   // EFFECTS: set data member a to _a and data member b to _b.
   //          Implementation omitted.
};

int main(){
   foo x(4);
   bar *pb = &x;
   pb->set(1);
   return 0;
}
```

4. (4 points) This snippet of code uses the STL container `deque`

```
deque<int> dq(10, 0);
deque<int>::const_iterator it;
for(it=dq.begin(); it<dq.end(); it++)
   cout << *it << " ";
int count = 0;
for(it=dq.begin(); it<dq.end(); it++){
   count ++;
   *it = count;
}
```

5. (4 points)

```
int add(int a, int b = 0, int c = 0, int d){
   return a + b + c + d;
}
int foo(int a, int d){
   return add(a, d);
}
```

6. (4 points) This code uses the STL containers `list` and `vector`:

```
list<int> l;
for(unsigned int i=0; i<10; i++){
   l.push_back(i);
```

```
}
list<int>:const_iterator it = l.begin();
unsigned int i = l.size()/3;
vector<unsigned int> v(it, it+i);
```

3 Functions and Inheritance (12 pts)

Provide your answers (1) to (12) on different lines of the file `Q3.txt`. Do not include extra empty lines between your answers. Your file should have 12 lines in total.

Given the following class declarations:

1. (4 points)
```
class A{
public:
    void f(){ cout << "A::f" << endl; }
    virtual void g(){ cout << "A::g" << endl; }
    virtual void h() = 0;
};

class B: public A{
public:
    void f(){ cout << "B::f" << endl; }
    void g(){ cout << "B::g" << endl; }
    void h(){ cout << "B::h" << endl; }
};

class C: public B{
public:
    virtual void f(){ cout << "C::f" << endl; }
    void h(){ B::h(); }
};

class D: public C {
public:
    void h(){ cout << "D::h" << endl; }
};
```

For each of the following commented line of code, write what is printed out at that line.

```
A *pa;
B b;
C c;
D d;

c.f();    // (1)
```

```
c.h();    // (2)

pa = &b;
pa->f();    // (3)
pa->g();    // (4)
pa->h();    // (5)

pa = &c;
pa->f();    // (6)
pa->g();    // (7)
pa->h();    // (8)

A &ra = d;
ra.g();     // (9)
ra.h();     // (10)

d.f();      // (11)
d.g();      // (12)
```

4 Dictionary (30 pts)

In this problem, you will implement some of the main methods of the dictionary ADT according to the templated class defined in the header file `dict.h`.

Here are some important instructions:

- **Do not modify** this header file.
- Provide your implementation in a file named `dict.hpp`.
- All the methods should be implemented in less than 15 lines. Actually, many methods can be implemented with very few lines.
- Do not include any other header files in your final submission.
- If you choose to implement any methods that are not marked by a TODO comment in the header file, do not include the extra implementation in your submission.

Not respecting those instructions may lead to lose all the points for this problem.

Here are some hints and suggestions:

- You may implement any non-member helper function if you want.
- In your implementation, you can use any methods of the STL containers.
- To use a companion type of templated class, you need to use the keyword `typename` before that type. For instance:

```
typename stack<T>::iterator it;
```

This is to tell the compiler to interpret `stack<T>::iterator` as a type.

- Think about the representation invariant before starting writing any code.

1. (30 points) Implement all the methods marked by a TODO comment in the header file.

5 Binary Tree (30 pts)

In this problem, you will implement some of the methods of a binary tree as defined in the header file `binaryTree.h`.

Here are some important instructions:

- **Do not modify** this header file.
- Provide your implementation in a file named `binaryTree.cpp`.
- All the methods should be implemented in less than 15 lines. Actually, many methods can be implemented with very few lines.
- Do not include any other header files in your final submission.
- If you choose to implement any methods that are not marked by a TODO comment in the header file, do not include the extra implementation in your submission.

Not respecting those instructions may lead to lose all the points for this problem.

Here are some hints and suggestions:

- You may implement any non-member helper function if you want.
- Think about the representation invariant before starting writing any code.

1. (30 points) Implement all the methods marked as TODO in the header file.