UTS IF2210/Pemrograman Berorientasi Objek
Rabu, 6 Maret 2019 Waktu: 100 menit
Hal. 1 dari 4

NIM: 13517012 Kelas : 03
Nama: Johanes

Soal 1. Kelas dan Inheritance

Diberikan header file **X.h**, **Y.h**, dan **Z.h** sebagai berikut.

1. **Lengkapilah** semua header file di bawah ini dengan *prototype* (tanpa membuat *inline implementation*) sesuai dengan spesifikasi yang dituliskan pada komentar. Gunakan nama fungsi/prosedur sesuai yang diberikan di spesifikasi. Tambahkan keyword *virtual*, *const*, dan *static* sesuai dengan kaidah yang diajarkan untuk menjamin kode Anda benar. Jika anggota kelas tidak perlu direalisasikan, tuliskanlah: **//Tak Perlu** (Anda juga tidak perlu menuliskan *prototype*-nya).
2. Tuliskan kode program **X.cpp, Y.cpp**, dan **Z.cpp** sesuai dengan spesifikasi pada header file, termasuk inisialisasi semua variabel *static*, pada halaman kosong di balik **halaman 1 dan 2** (spesifikasi tidak perlu ditulis ulang).

```
// File: X.h
#ifndef _X_H_
#define _X_H_

class X {
    protected:
        int s;
        int a;
        int * tabdata;
        static int nbX; // banyaknya objek X yang sedang hidup pada suatu saat
    public:
        // default ctor: set nilai atribut sbb: s = 10; a = 0
        // alokasi tabdata dg ukuran s, dan inisialisasi semua elemen tabdata dg nilai = 0
        X();
        // user-defined ctor: set nilai atribut sbb: s = _s; a = _a
        // alokasi tabdata dg ukuran s, dan inisialisasi semua elemen tabdata dg nilai = _a
        X(int _s, int _a);
        // cctor: pastikan semua elemen tabdata tersalin dengan baik
        X(const X& x);
        // dtor: pastikan memori yang digunakan tabdata dibebaskan
        ~X();
        // operator assignment: pastikan semua elemen tabdata tersalin dengan baik
        X& operator=(const X& x);
        // getS: menghasilkan nilai atribut s
        int getS() const;
        // getA: menghasilkan nilai atribut a
        int getA() const;
        // getEl: menghasilkan nilai elemen atribut tabdata pada indeks ke-i
        int getEl(int i) const;
        // setS: mengubah nilai atribut s dg nilai baru
        void setS(int _s);
        // setA: mengubah nilai atribut a dg nilai baru
        void setA(int _a);
        // setEl: mengubah nilai elemen atribut tabdata pada indeks ke-i dg nilai baru
        void setEl(int i, int x);
        // print: mencetak nilai s, a, dan semua nilai elemen pada tabdata (format bebas)
        void print();
        // fx: menerima masukan sebuah integer, misalnya h
        // melakukan perubahan terhadap setiap elemen tabdata dg aturan yg tergantung pada
        // jenis objek X (belum bisa ditentukan di kelas ini)
        virtual void fx(int h) = 0;
        // getNbX: menghasilkan nilai nbX
        static int getNbX() const;
};
#endif
```

```
// File: Y.h
#ifndef _Y_H_
#define _Y_H_

#include "X.h"

class Y : public X {
    private:
        int c;
    public:
        // default ctor: set nilai atribut a, s, dan tabdata spt pd default ctor X
        // dan set atribut c = 1
        Y();
        // user-defined ctor: set nilai atribut a, s, dan tabdata spt pd user-defined ctor X
        // dan set atribut c = _c
        Y(int _s, int _a, int _c);
        // cctor Y
        // Tdk perlu
        // dtor Y
        // Tdk Perlu
        // operator assignment Y
        // Tdk perlu.
        // getC: menghasilkan nilai atribut C
        int getC() const;
        // setC: mengubah nilai atribut c dg nilai baru
        void setC(int _c);
        // print: mencetak nilai s, a, dan semua nilai elemen pada tabdata dgn format sama spt pd X;
        // ditambah mencetak nilai c (format bebas)
        void Print();
        // fx: menerima masukan sebuah integer, misalnya h
        // melakukan perubahan terhadap setiap elemen tabdata dg aturan sbb:
        // misal nilai elemen ke-i adalah e[i]; maka nilai e[i] diubah mjd = e[i] * (c + h)
        void fx(int h);
};
#endif
```

```
// File: Z.h
#ifndef _Z_H_
#define _Z_H_

#include "X.h"

class Z : public X {
    public:
        // default ctor: set nilai atribut a, s, dan tabdata spt pd default ctor X
        // Tdk perlu
        // user-defined ctor: set nilai atribut a, s, dan tabdata spt pd user-defined ctor X
        // Tdk perlu
        // cctor Z
        // Tdk perlu
        // dtor Z
        // Tdk perlu
        // operator assignment Z
        // Tdk perlu
        // print: mencetak nilai s, a, dan semua nilai elemen pada tabdata dgn format sama spt pada X
        // Tdk Perlu
        // fx: menerima masukan sebuah integer, misalnya h
        // melakukan perubahan terhadap setiap elemen tabdata dg aturan sbb:
        // misal nilai elemen ke-i adalah e[i]; maka nilai e[i] diubah mjd = e[i] * h
        void fx(int h);
};
#endif
```

NIM: 13519012 Kelas : 03
Nama: Johanes

Soal 2. *Template* Kelas Dictionary

Diberikan *template* kelas entry sebagai berikut.

```
template<class K, class V>
class entry {
    private:
       K key;
       V value;
    public:
       entry(K k, V v): key(k), value(v) {}
       K getKey() { return key; }
       V getValue() { return value; }
       void setValue (V newValue) { value = newValue; }
};
```

Buatlah *template* kelas dictionary yang dapat berisi sekumpulan entry. Berikut contoh penggunaan *template* kelas dictionary. Implementasi *template* harus mencakup semua fungsi anggota (publik) yang pada contoh di bawah ditandai dengan cetak tebal dan garis bawah. Definisikan *exception* jika diminta.

```
int main() {
    dictionary<int,string> d; // dapat berisi sekumpulan entry<int,string>

    try {
        d.put(1,"one"); // isi d saat ini: [1:one]                    (menyimpan entry(1,"one") ke d)
        cout<<1<<": "<<d.get(1)<<endl; // mencetak "1: one"   (mengambil value entry yang memiliki key 1)

        d.put(2,"two"); // isi d saat ini: [1:one,2:two]
        cout<<1<<": "<<d.get(1)<<endl; // mencetak "1: one"
        cout<<2<<": "<<d.get(2)<<endl; // mencetak "2: two"

        d.put(1,"satu");  // isi d saat ini: [1:satu,2:two],     (menimpa value dari key 1 dengan "satu")
                          // BUKAN [1:one,2:two,1:satu]
        d.put(3,"three"); // isi d saat ini: [1:satu,2:two,3:three]
        cout<<1<<": "<<d.get(1)<<endl; // mencetak "1: satu"
        cout<<2<<": "<<d.get(2)<<endl; // mencetak "2: two"
        cout<<3<<": "<<d.get(3)<<endl; // mencetak "3: three"

        d.remove(2); // isi d saat ini: [1:satu,3:three]          (menghapus entry yang memiliki key 2)
        cout<<1<<": "<<d.get(1)<<endl; // mencetak "1: satu"
        if (d.containsKey(2)) {              // (mengembalikan true jika d memiliki entry dengan key 2)
            cout<<2<<": "<<d.get(2)<<endl; // tidak ada output
        }
        cout<<2<<": "<<d.get(2)<<endl; // mencetak "exception: key not found!"     (key 2 tidak ada di d)
        cout<<3<<": "<<d.get(3)<<endl; // tidak ada output karena sudah exception
    } catch (const char* c) {
        cout<<"exception: "<<c<<endl;
        return 1;
    }

    return 0;
}
```

Petunjuk:

- Anda tidak perlu membuat *method* 4 sekawan (dianggap sudah ada).
- Secara internal, Anda boleh menentukan sendiri bagaimana kumpulan entry disimpan (*array*, std::vector, dsb.). Petunjuk: jika menggunakan vector Anda tidak perlu memikirkan penggeseran elemen *array* bilamana terdapat elemen yang dihapus di tengah *array* (gunakan fungsi erase()).

Tuliskan jawaban di sisa halaman ini dan di halaman kosong di baliknya.

NIM: 13519012 Kelas : 03
Nama: Johanes

Soal 3. Reading Comprehension

Diberikan program sebagai berikut, tuliskan hasil eksekusinya di balik halaman 4. Program dijamin lolos kompilasi.

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

class A {
   public:
      A(const char i=0) {
         cout << "A::ctor" << endl;
         x = i;
      }
      A(const A& i) {
         cout << "A::cctor" << endl;
         x = i.x;
      }
      ~A() {
         cout << "A::dtor" << endl;
      }
      A& operator=(const A& i) {
         cout << "A::opr=" << endl;
         x = i.x;
         return *this;
      }
      void f() {
         cout << "A::f()" << endl;
      }

   private:
      char x;
};

class B {
   public:
      B(const int n) {
         cout << "B::ctor" << endl;
         pa = new A[n];
         this->n = n;
      }
      B(const B& a) {
         cout << "B::cctor" << endl;
         n = a.n;
         pa = new A[n];
      }
      B& operator=(const B& b) {
         cout << "B::opr=" << endl;
         if (this != &b) {
            delete[] pa;
            n = b.n;
            pa = new A[n];
         }
         return *this;
      }
      virtual ~B() {
         cout << "B::dtor" << endl;
         delete[] pa;
      }
      virtual void f() {
         cout << "B::f()" << endl;
         for (int i=0; i<n; i++)
            pa[i].f();
      }

   private:
      A *pa;
      int n;
};
```

```
class C: public A, public B {
   public:
      C(const char n): B(n) {
         cout << "C::ctor" << endl;
      }

      virtual ~C() {
         cout << "C::dtor" << endl;
      }

      virtual void f() {
         cout << "C::f()" << endl;
         A::f();
         B::f();
      }
};

int main() {

   cout << "1:" << endl;
   A *pa;
   B *pb;
   C *pc = new C(3);

   cout << "2:" << endl;
   pa = pc;
   pb = pc;
   pa->f();
   pb->f();

   cout << "3:" << endl;
   pc->B::f();

   cout << "4:" << endl;
   delete pb;

   cout << "5:" << endl;
   C c1(2);
   C c2 = c1;

   cout << "6:" << endl;
   c2 = c1;

   cout << "0:" << endl;

   return 0;
}
```

UTS SEMESTER II – 2020/2021

Section 1

Soal di Quizizz yang tidak sempat didokumentasikan

Section 2

Soal 1. Inheritance, Abstract Class/Function

A. Lengkapi header kelas abstrak Property berikut ini dengan *prototype* (tanpa membuat *inline implementation*) sesuai dengan spesifikasi yang dituliskan pada komentar. Gunakan nama fungsi/prosedur sesuai yang diberikan di spesifikasi. Tambahkan *keyword* virtual, const, dan/atau static sesuai dengan kaidah yang diajarkan untuk menjamin kode Anda benar. Jika anggota kelas tidak perlu direalisasikan, tuliskanlah: // TIDAK PERLU (Anda juga tidak perlu menuliskan *prototype*-nya).

```
// Class Property
// File: Property.h

#ifndef PROPERTY_H
#define PROPERTY_H

#include <string>
using namespace std;

class Property {
  protected:
  // Atribut
    string name;
    string type; //hotel; hostels; villas; cottages
    int openYear;
  public:
    // User-defined constructor: set nilai atribut berdasarkan nilai
    // parameter masukan

    // Default constructor: set nilai atribut sbb:
    // name = "noname"; openYear = 1900; type = "none"

    // Copy constructor

    // Destructor

    // Operator Assignment

    // ... set_name(...)

    // ... get_name()

    // ... set_type(...)

    // ... get_type()

    // ... get_age(), asumsikan tahun saat ini dapat diakses dengan
```

```
    // makro CURRENT_YEAR

    // ... displayInfo(): Mencetak nama, umur, type, dan rate Property

    // ... rate(): menghitung biaya property sesuai dengan umur dan
    // tergantung type property

};

#endif // PROPERTY_H
```

B. Buatlah *subclass* dengan mewariskan kelas `Property` pada kelas `Hotel` dan `Hostel`, dengan ketentuan sebagai berikut (buat *overload function* jika diperlukan):
   a. Kelas `Hotel` mencatat jumlah star setiap Hotel, diakses menggunakan `set_star()` dan `get_star()`. Override `displayInfo()` untuk mencetak informasi tambahan yaitu *total rate* yang didapatkan dari perkalian antara rate dengan star.
   b. Kelas `Hostel` mencatat jumlah facility yang sudah digunakan (`set_facility()`, `get_facility()`) dan memiliki fungsi `calculateFacility()` yang menghasilkan *expense* yaitu jumlah facility dikalikan dengan 80000. *Override* `displayInfo()` untuk mencetak informasi tambahan yaitu *expense*.

C. Tuliskan implementasi kelas abstrak `Property` dalam `Property.cpp` sesuai dengan spesifikasi pada *header file*.

D. Lengkapi `main.cpp` berikut pada bagian yang kosong untuk menguji perilaku kelas-kelas yang Anda buat dengan:

```
#include "Property.h"
#include <iostream>
using namespace std;

int main() {
  ____________ hilton = Hotel("Hotel Hilton", 2000);
  ____________ vio = Hostel("Hostel Vio", 2003);
  // cetak semua property dan informasi semua property
  // [gunakan Property::displayInfo()]
    ____________________________________
    ____________________________________

}
```

Soal 2. Queue dan PriorityQueue

Queue adalah sebuah struktur data yang dapat menyimpan sekumpulan elemen, dimana sebuah elemen dimasukkan dan dikeluarkan secara *first-in-first-out*. Sebuah queue umumnya memiliki dua method penting, yaitu *enqueue* untuk memasukkan elemen ke posisi paling akhir, dan *dequeue* untuk mengeluarkan elemen pada posisi paling depan. Berikut ini adalah contoh serangkaian aksi *enqueue* dan *dequeue* pada sebuah queue dengan elemen bertipe integer dan memiliki kapasitas 10.

No.	Aksi	Kondisi Queue
0	Kondisi awal kosong	Elemen:

1	enqueue(8)	Elemen: 8
2	enqueue(6)	Elemen: 8, 6
3	enqueue(4)	Elemen: 8, 6, 4
4	dequeue(), returns 8	Elemen: 6, 4
5	enqueue(2)	Elemen: 6, 4, 2
6	dequeue(), returns 6	Elemen: 4, 2

Struktur queue ini dapat di-*extend* lebih lanjut menjadi apa yang disebut PriorityQueue. Pada PriorityQueue, setiap elemen memiliki prioritas yang menentukan posisi elemen tersebut pada queue. Semakin tinggi prioritas sebuah elemen, akan semakin di depan posisinya. Elemen-elemen dengan prioritas yang sama, akan diperlakukan secara *first-in-first-out* sebagaimana pada Queue biasa. Berikut ini adalah contoh serangkaian aksi *enqueue* dan *dequeue* pada sebuah PriorityQueue dengan elemen bertipe integer dan memiliki kapasitas 10. Fungsi *enqueue* memiliki dua parameter, yaitu enqueue(elemen, prioritas).

No.	Aksi	Kondisi Queue
0	Kondisi awal kosong	Elemen: Prioritas:
1	enqueue(8, 1)	Elemen: 8 Prioritas: 1
2	enqueue(6, 2)	Elemen: 6, 8 Prioritas: 2, 1
3	enqueue(4, 1)	Elemen: 6, 8, 4 Prioritas: 2, 1, 1
4	dequeue(), returns 6	Elemen: 8, 4 Prioritas: 1, 1
5	enqueue(2, 3)	Elemen: 2, 8, 4 Prioritas: 3, 1, 1
6	dequeue(), returns 2	Elemen: 8, 4 Prioritas: 1, 1

Sebuah struktur data Queue dan PriorityQueue dapat dibuat generik sehingga elemennya dapat bertipe apapun. Dengan bahasa C++,

A. Buatlah sebuah kelas Queue generik dengan kapasitas maksimal 10 elemen.
B. Turunkan kelas Queue tersebut menjadi sebuah kelas `PriorityQueue` generik.

   Kelas-kelas tersebut minimal harus memiliki:

a. Default constructor
b. Destruktor
c. Method enqueue, untuk memasukkan sebuah elemen ke dalam queue. Method ini akan melempar sebuah exception jika queue sudah penuh.
d. Method dequeue, untuk mengeluarkan sebuah elemen ke dalam queue. Method ini akan melempar sebuah exception jika queue kosong.
e. Method `print`, untuk menampilkan isi queue (bentuk tampilan bebas)

C. Tulislah juga sebuah *main function* yang membuat sebuah `Queue` dan `PriorityQueue` dengan isi seperti contoh di atas. Gunakanlah *exception handler* di *main function* untuk menangani *exception* yang dilemparkan oleh kelas `Queue` dan `PriorityQueue`.

Anda boleh menggunakan kerangka program berikut ini. Jika ada bagian kerangka program yang kurang lengkap atau salah, Anda dapat melengkapinya atau menggantinya.

```
#include <iostream>

using namespace std;

template <class T>
class Queue {
  private:
    T *elements;
    int nElements;
  public:
    Queue();
    ~Queue();
    void enqueue(const T&);
    T dequeue();
    void print();
};

// Default constructor

// Destructor

// Method enqueue

// Method dequeue

// Method print


template <class T>
class PriorityQueue : public Queue<T> {
  private:
    int *priorities;
  public:
    PriorityQueue();
    ~PriorityQueue();
    void enqueue(const T&, int);
};
```

```
// Default constructor

// Destructor

// Method enqueue

// Method dequeue, override if needed

// Method print, override if needed


// Main function
int main() {

}
```

Soal 3. STL

Pada Python, kita dapat membuat *tuple*, yakni gabungan dari beberapa nilai, misalnya (123, "abc", true). Di C++, kita juga dapat menggunakan STL `pair<>` untuk menggabungkan dua nilai. Namun, kali ini Anda membutuhkan tuple untuk tiga nilai. Karena itu, buatlah sebuah kelas bernama `Triplet` yang dapat menampung tiga nilai dengan tipe data yang berbeda.

Contoh Kode	Contoh Output

`Triplet<int, int, int> a(3, 2, -5);`
 	
`cout << a.getFirst() << endl;`
`cout << a.getSecond() << endl;`
`cout << a.getThird() << endl;`
	`3`
	`2`
	`-5`

`Triplet<string, float, string> a("abc", 4.5, "def");`
`Triplet<string, float, string> b("ghi", -1.0, "def");`
`Triplet<string, float, string> c("abc", 4.5, "def");`
 	
`if (a == b) {`
`  cout << "a == b" << endl;`
`}`
`if (a == c) {`
`  cout << "a == c" << endl;`
`}`
	`a == c`

SECTION I: SOAL KONSEP

1. Bila kita memerlukan fungsi yang melakukan operasi sejenis terhadap tipe data yang berbeda, maka kita menggunakan:
   a. virtual function
   b. template (/generic) function
   c. friend function
   d. const function
   e. cctor
2. (Multiple Answer)Berikut ini yang merupakan jenis-jenis inheritance pada C++ adalah
   a. Hierarchical inheritance
   b. Double inheritance
   c. Hybrid Virtual inheritance
   d. Multilevel inheritance
3. (Multiple Answer)Jika kelas D1 dan D2 merupakan turunan dari kelas B yang mengandung definisi method virtual M, dan kelas E yang diturunkan dari D1 dan D2 tidak meng-override M, maka:
   a. Salah satu dari D1 atau D2 harus meng-override M
   b. D1 dan D2 keduanya harus meng-override M
   c. Salah satu dari D1 atau D2 boleh meng-override M
   d. D1 dan D2 tidak perlu meng-override M
4. (Multiple Answer)Mana sajakah pernyataan yang benar terkait exception handling di C++
   a. Bekerja dengan mengubah alur eksekusi program sambil mengembalikan objek tertentu sebagai informasi alur yang baru
   b. Jika sebuah method ditulis menangani exception, invokasi sebaiknya dilakukan dalam sebuah blok try ... finally hrsnya try ... catch
   c. Error selalu harus ditangani dengan exception handling ga selalu
   d. Exception dapat menyebabkan program terminate abnormally
5. (Multiple Answer)Diberikan screenshot program terlampir. Pilih pernyataan yang benar:

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

class Kendaraan {
public:
    virtual void info(){
        cout << "Ini adalah kendaraan."<< endl;
    }
};

class Mobil:virtual public Kendaraan {
public:
    void info(){
        cout << "Ini adalah kendaraan mobil."<< endl;
    }
};

class Motor:virtual public Kendaraan {
 public:
    void info(){
        cout << "Ini adalah kendaraan motor."<< endl;
    }
};

class Batmobile:public Mobil, public Motor {
public:
    void info(){
        cout << "Ini adalah kendaraan mobil motor Batmobile."<< endl;
    }
};

int main(){
    Kendaraan kendaraan;
    Mobil mobil;
    Motor motor;
    Batmobile batmobile;

    Kendaraan* obj1 = &kendaraan;
    obj1->info();
    obj1 = &mobil;
    obj1->info();
    obj1 = &motor;
    obj1->info();
    obj1 = &batmobile;
    obj1->info();

    Kendaraan obj2 = kendaraan;
    obj2.info();
    obj2 = mobil;
    obj2.info();
    obj2 = motor;
    obj2.info();
    obj2 = batmobile;
    obj2.info();

    return 0;
}
```

a. Baris ke-41 dan ke-43 akan menampilkan string "Ini adalah kendaraan mobil." dan "Ini adalah kendaraan motor.".
b. Baris ke-45 menyebabkan program error.
c. Baris ke-45 akan menampilkan string "Ini adalah kendaraan.".
d. Baris ke-54 menyebabkan program error.
e. Baris ke-54 akan menampilkan string "Ini adalah kendaraan.".

6. (Multiple Answer)Berikut ini merupakan prinsip dari OOP
   a. Monomorfisme poly
   b. Encapsulation
   c. Abstraction
   d. Inheritance

7. Container dengan struktur sekuens berindeks yang memperbolehkan menambah dan menghapus elemen di awal dan di akhir disebut sebagai
    a. iterator
    b. deque
    c. queue
    d. bidirectional
8. (Multiple Answer)Yang harus ada pada sistem pemrograman berorientasi objek
    a. Objek
    b. Kelas
    c. Method
    d. Inheritance
9. (Multiple Answer)Berikut adalah karakteristik kelas abstrak (abstract base class/ABC)
    a. Kita tidak dapat membuat objek dari kelas ABC
    b. Kelas ABC dapat langsung diinstansiasi gabisa
    c. Objek yang diinstansiasi terbatas pada instansiasi kelas turunan ABC
    d. Kelas ABC setidaknya memiliki satu method yang tidak diimplementasikan
    e. Suatu kelas dikatakan kelas ABC jika terdapat method yang dioverride oleh kelas turunannya pure virtual(?)
10. (Multiple Answer)Manakah penjelasan yang tepat untuk jenis iterator berikut
    a. Output: iterator yang menulis elemen dari container dan bisa bergerak maju atau mundur hny bisa maju
    b. Forward: iterator untuk membaca/menulis elemen dari container, hanya bisa bergerak maju, tapi bisa multipass mempertahankan posisi multipass, bisa lewat 2 kali, cm bs maju
    c. Random access: iterator yang dapat maju, tetapi bisa juga mundur, dan juga lompat ke elemen manapun
    d. Input: iterator yang membaca elemen dari container dan hanya bisa bergerak maju
11. (Multiple Answer)Pilihlah pernyataan yang benar mengenai beberapa jenis objek dalam program C++
    a. Free Store Object
    b. Member Object
    c. Dynamic Object (?)
    d. Dependent Object
12. (Multiple Answer)Jika kita membutuhkan kelas Stack yang ketika setiap diinstansiasi dapat menampung tipe data yang berbeda, maka kita bisa memanfaatkan:
    a. generic class
    b. standard template library
    c. polymorphism dari base class Array
    d. struct
    e. operator overloading

SECTION II: SOAL PEMROGRAMAN

Soal 1. Inheritance, Abstract Class/Function

a. Lengkapi header kelas abstrak `Hotel` berikut ini dengan prototype (tanpa membuat *inline implementation*) sesuai dengan spesifikasi yang dituliskan pada komentar. Gunakan nama fungsi/prosedur sesuai yang diberikan di spesifikasi. Tambahkan keyword `virtual`, `const`, dan/atau `static` sesuai dengan kaidah yang diajarkan untuk menjamin kode Anda benar. Jika anggota kelas tidak perlu direalisasikan, tuliskanlah:

`// TIDAK PERLU` (Anda juga tidak perlu menuliskan *prototype*-nya).

```
// Class Hotel
// File: Hotel.h

#ifndef HOTEL_H
#define HOTEL_H

#include <string>
using namespace std;

class Hotel {
  protected:
  // Atribut
    string name;
    string bintang; //Melati; bintang_tiga; bintang_empat; bintang_lima
    int openYear;
  public:
    // User-defined constructor: set nilai atribut berdasarkan nilai parameter masukan

    // Default constructor: set nilai atribut sbb:
    // name = "noname"; openYear = 1900; bintang = "none"

    // Copy constructor

    // Destructor

    // Operator Assignment

    // ... set_name(...)

    // ... get_name()

    // ... set_bintang(...)

    // ... get_bintang()

    // ... get_age(), asumsikan tahun saat ini dapat diakses dengan makro CURRENT_YEAR

    // ... displayInfo(): Mencetak nama, umur hotel, bintang, dan room_rate

    // ... rate(): menghitung biaya menginap sesuai dengan umur dan tergantung bintang hotel


};

#endif // HOTEL_H
```

b. Buatlah *subclass* dengan mewariskan kelas `Hotel` pada kelas `bintang_empat` dan `bintang_lima`, dengan ketentuan sebagai berikut (buat *overload function* jika diperlukan):
   - Kelas `bintang_empat` mencatat jumlah star layanan setiap hotel bintang empat, diakses menggunakan `set_star()` dan `get_star()`. *Override* `displayInfo()` untuk mencetak informasi tambahan yaitu totalRate yang didapatkan dari perkalian antara rate dengan star.
   - Kelas `bintang_lima` mencatat jumlah facility yang sudah digunakan (`set_facility()`, `get_facility()`) dan memiliki fungsi `calculateFacility()` yang menghasilkan expense yaitu jumlah facility dikalikan dengan 100000. *Override* `displayInfo()` untuk mencetak informasi tambahan yaitu expense.

c. Tuliskan implementasi kelas abstrak `Hotel` dalam `Hotel.cpp` sesuai dengan spesifikasi pada header file.

d. Lengkapi `main.cpp` berikut pada bagian yang kosong untuk menguji perilaku kelas-kelas yang Anda buat dengan:

```
#include "Hotel.h"
#include <iostream>
using namespace std;

int main() {

    _____ aston = bintang_empat("Hotel Aston", 2010);

    _____ padma = bintang_lima("Hotel Padma", 2000);

    // cetak semua hotel dan informasi semua hotel
    // [gunakan Hotel::displayInfo()]

        _____

        _____


}
```

Soal 2. Generic Class dan Exception Handling

Berikut ini adalah contoh serangkaian aksi enqueue dan dequeue pada sebuah queue dengan elemen bertipe integer dan memiliki kapasitas 3 elemen.

No.	Aksi	Kondisi Queue
0	Kondisi awal kosong	`Elemen:`
1	dequeue()	`Elemen:` `Queue telah kosong`
2	enqueue(8)	`Elemen: 8`
3	enqueue(6)	`Elemen: 8, 6`
4	enqueue(4)	`Elemen: 8, 6, 4`
5	enqueue(9)	`Elemen: 8, 6, 4` `Queue telah penuh`
6	dequeue(), returns 8	`Elemen: 6, 4`
7	enqueue(2)	`Elemen: 6, 4, 2`
8	dequeue(), returns 6	`Elemen: 4, 2`

Sebuah struktur data Queue dapat dibuat generik sehingga elemennya dapat bertipe apapun.

Tugas kalian adalah:

1. Dengan bahasa C++, buatlah sebuah kelas Queue generik dengan kapasitas maksimal 3 elemen. Kelas tersebut minimal harus memiliki:

    a. Default constructor.
    b. Destructor.
    c. Method enqueue, untuk memasukkan sebuah elemen ke dalam Queue. Method ini akan melempar sebuah exception jika Queue sudah penuh.
    d. Method dequeue, untuk mengeluarkan sebuah elemen ke dalam Queue. Method ini akan melempar sebuah exception jika Queue kosong.
    e. Operator overloading << untuk menampilkan isi Queue (bentuk tampilan bebas).

2. Membuat kelas abstrak QueueException dengan method what() yang mengembalikan informasi tipe exception. Kelas turunan QueueException dapat menangani 2 jenis exceptions:

    a. Suatu Queue telah kosong.
    b. Suatu Queue telah penuh.

3. Tulislah main program yang membuat sebuah Queue dengan isi seperti contoh di atas. Gunakanlah exception handler untuk menangani exception yang dilemparkan oleh kelas Queue.
4. Tunjukkan bahwa struktur data Queue telah dibuat generik dengan mampu menampung elemen yang beragam:

    a. Minimal 2 tipe data “built-in”, yakni Queue of int dan Queue of float.
    b. Minimal 1 tipe data “customized”, yakni Queue of Mobil (definisi kelas Mobil dibebaskan).

Gunakan kerangka program berikut ini. Program dikerjakan dalam satu buah file main.cpp untuk sekedar alasan kepraktisan dalam pemeriksaan ujian. Jika ada bagian kerangka program yang kurang lengkap atau salah, Anda dapat melengkapinya atau menggantinya.

```cpp
#include <iostream>

using namespace std;

template <class T>
class Queue {
  private:
    // lengkapi private member di sini
    T *elements;
    ...
  public:
  // lengkapi public member di sini
   ...

};

// Default constructor
// Destructor
// Method enqueue
// Method dequeue
// Operator overloading <<
```

```cpp
// Main function
int main() {



}
```

Soal 3. STL

Di negara NeverHeardBefore, pemimpin pemerintahan memiliki masa jabatan yang ditentukan oleh voting. Seorang pemimpin memiliki nilai diri tertinggi dibandingkan dengan birokrat lainnya. Setiap pemimpin/ birokrat memiliki dua birokrat lainnya untuk membantu, dengan ketentuan nilai diri mereka lebih tinggi dari dua asistennya tersebut. Struktur seperti ini dikenal dengan max heap yang merupakan implementasi dari priority queue. Jika pemimpin sekarang di-*vote* untuk berhenti memimpin, maka penggantinya diambil dari salah satu asisten yang memiliki skor tertinggi. Gunakan STL untuk mengimplementasikan hal berikut:

1. Membuat sebuah heap yang terdiri dari 50 bilangan integer random untuk merepresentasikan struktur pemimpin/ birokrat negara NeverHeardBefore
2. Menambahkan birokrat baru dengan nilai diri integer random
3. Melakukan vote terhadap pemimpin (menghapus elemen maksimum dari heap)
4. Mengurutkan struktur birokrat/ heap tersebut.

Note: kalau butuh contoh heap http://www.cplusplus.com/reference/queue/priority_queue/

atau dapat juga memanfaatkan vector