Uno.Holmer@chl.chalmers.se

Lösningsförslag till tentamen* *Preliminär*

Kursnamn	**Objektorienterad programutveckling**
Tentamensdatum	**2001-10-22**
Program	**DAI 2**
Läsår	**2001/2002, lp I/II**
Examinator	**Uno Holmer**

* se även `www.chl.chalmers.se/~holmer/` där finns det mesta av kursmaterialet.

Uppgift 1 (10 p)

Ingen lösning ges. Se kurslitteraturen.

Uppgift 2 (12 p)

```
class String : public Vector<char> {
public:
    explicit String( int size = 0 ) : Vector<char>( size ) {}
    String operator+ ( const String & rhs ) const;
    friend istream & operator>> ( istream & in, String & rhs );
    friend ostream & operator<< ( ostream & out, const String & rhs );
};

String String::operator+ ( const String & rhs ) const {
    String tmp = *this;
    for ( int i = 0; i < rhs.size(); i++ )
        tmp.push_back( rhs[ i ] );
    return tmp;
}

istream & operator>> ( istream & in, String & rhs ) {
    rhs.resize( 0 );
    char c;
    in >> ws;
    while ( ( c = in.get() ) != ' ' && c != '\n' )
        rhs.push_back( c );
    return in;
}

ostream & operator<< ( ostream & out, const String & rhs ) {
    for ( int i = 0; i < rhs.size(); i++ )
        out << rhs[ i ];
    return out;
}
```

Uppgift 3 (1+6+3 p)

a) (1 p)

```
class Greater {
public:
    bool operator() ( int x, int y ) { return x > y; }
};
```

b) (6 p)

```
template <class T, class Greater>
Cref<T> max( const Vector<T> & v, Greater gt ) {
    if ( v.empty() )
        return Cref<T>();

    int maxPos = 0;
    for ( int i = 1; i < v.size(); i++ )
        if ( gt( v[ i ], v[ maxPos ] ) )
            maxPos = i;
    return Cref<T>( v[ maxPos ] );
}
```

c) (3 p)

```
Cref<int> m = max( v, Greater() );
if ( ! m.isNull() )
    cout << m.get() << endl;
else
    cout << "max undefied for empty vector" << endl;
```

Uppgift 4 (7+5 p)

a) (6 p)

De tre första är uppenbara eftersom `a` är deklarerad av typ `A`:

1: `A::f`

2: `A::g`

3: `A::h`, att `h` är virtuell saknar betydelse här

Pekaren `ap` har statisk typ `*A` och får dynamisk typ `*B` när den tilldelas adressen till variabeln `b`.

4, 5: `A::f` resp. `A::g`, `f` och `g` är ej virtuella varför `ap`:s statiska typ `*A` bestämmer vilka funktioner som anropas.

6: `B::h`, `h` är virtuell och anropas med dynamisk bindning. Eftersom den är omdefinierad i `B` anropas `B::h`.

7: Kompileringsfel, funktionen `i` är ej deklarerad i `A`.

b) (5 p)

Diagram a beskriver en möjlig instansiering av såväl klassdiagram 1 som 2. Diagram b beskriver instanser av 2. Däremot går inte b ihop med 1 eftersom b beskriver en situation där objekten `g3` och `g4` delar objektet `d3`, men av b framgår ju att `G` har en aggregatrelation till övriga klasser och därmed förfogar exklusivt över sina komponenter, vilket strider mot delning.

Uppgift 5 (3+6+4+3 p)

a) (3 p)

```
class ServiceReminder : public Alarm {
public:
    ServiceReminder( const char *partId ) : Alarm(partId) {}
    void what() const {
        when().print();
        cout << ": time for service of the " << where() << endl;
    }
};
```

b) (6 p)

```
class TemperatureAlert : public Alarm {
public:
    TemperatureAlert( const char *where, int aLimit,
                      int aTemperature )
        : Alarm(where), theTemperatureLimit(aLimit),
          theTemperature(aTemperature) {}

    void what() const {
        when().print();
        cout << ": the temperature in the "
             << where() << " was ";
        int tempDiff = theTemperature - theTemperatureLimit;
        if ( tempDiff >= 0  )
            cout << tempDiff << " degrees above the limit "
                 << theTemperatureLimit<< endl;
        else
            cout << -tempDiff << " degrees below the limit "
                 << theTemperatureLimit<< endl;
    }
private:
    int theTemperatureLimit, theTemperature;
};
```

c) (4 p)

```
class AlarmLog {
public:
    void log( Alarm *anAlarm );
    void printLog() const;
private:
    Vector<Alarm*> theAlarmLog;
};

void AlarmLog::log( Alarm *anAlarm ) {
    theAlarmLog.push_back( anAlarm );
}

void AlarmLog::printLog() const {
    for ( int i = 0; i < theAlarmLog.size(); i++ )
        theAlarmLog[ i ]->what();
}
```

d) (3 p)

```
AlarmLog aLog;
aLog.log( new TemperatureAlert( "engine", 95, 70 ) );
aLog.log( new ServiceReminder( "gear box" ) );
```

```
aLog.log( new TemperatureAlert( "steam boiler", 100, 105 ) );
aLog.log( new CoffeeBreak( "Cake is served in the lounge" ) );
aLog.printLog();
```