Lösningsförslag till tentamen *

Kursnamn	**Programmeringsteknik D, fk (3p)**
Tentamensdatum	**2001-05-19**
Program	**DAI, åk 1**
Läsår	**2000/2001, lp IV**
Examinator	**Uno Holmer**

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

Uppgift 1 (10 p)

Se kurslitteraturen.

Uppgift 2 (9+4 p)

a) (9 p)

```
class CupOfDice {
  ...
private:
    Die **theDice;
    int noOfDice;
    int capacity;
};

CupOfDice::CupOfDice() : noOfDice(0), capacity(1) {
    theDice = new Die*[ capacity ];
}

CupOfDice::~CupOfDice() {
    for ( int i = 0; i < noOfDice; i++ )
        delete theDice[ i ];
    delete [] theDice;
}

CupOfDice & CupOfDice::operator+=( Die *d ) {
    if ( noOfDice == capacity ) {
        Die **old = theDice;
        theDice = new Die*[ capacity *= 2 ];
        for ( int i = 0; i < noOfDice; i++ )
            theDice[ i ] = old[ i ];
        delete old;
    }
    theDice[ noOfDice++ ] = d;
    return *this;
}

void CupOfDice::shake() {
    for ( int i = 0; i < noOfDice; i++ )
        theDice[ i ]->toss();
}
```

```
int CupOfDice::getSum() const {
    int sum = 0;
    for ( int i = 0; i < noOfDice; i++ )
        sum += theDice[ i ]->getValue();
    return sum;
}
```

b) (4 p)

```
void main() {
    randomize();
    CupOfDice cd;
    // Make 5 random dice with 2 to 10 sides
    for ( int i = 0; i < 5; i++ )
        cd += new Die( 2 + random( 9 ) );
    cout << "Antal kast: ";
    int n;
    cin >> n;
    long int sum = 0;
    for ( int i = 0; i < n; i++ ) {
        sum += cd.getSum();
        cd.shake();
    }
    cout << sum/float(n) << endl;
}
```

Uppgift 3 (7 p)

Följande utskrifter kan ske: " Some error", "G returned 246", "456", "G returned 999".
Motivering:

- Om random returnerar 0 i F kastas det booleska undantaget false vilket fångas först i huvudprogrammets defaultfångare som skriver ut "Some error".
- Om random returnerar 1 kastas heltalsundantaget 123 vilket fångas i G som returnerar 246 till main som skriver ut "G returned 246".
- Om random returnerar 2 kastas strängundantaget "Positive" vilket fångas i G som återkastar heltalsundantaget 456 som fångas i main som skriver ut "456".
- Om random returnerar 3 returnerar F ett tal i intervallet [-5,4].
  - Är talet negativt kastar G strängundantaget "Negative" som fångas i G:S egen hanterare som returnerar 246 till main som skriver ut "G returned 246".
  - Är talet positivt returnerar G 999 till main som skriver ut "G returned 999".

Uppgift 4 (10 p)

Man kan tänka sig åtminstone fyra varianter av denna funktionsmall.

Alt. 1 Returnera en kopia av det största elementet.

```
template <class T>
T max( T a[], int n ) throw ( std::range_error ) {
    if ( n < 1 )
        throw std::range_error( "max: invalid range" );
    T m = a[ 0 ];
    for ( int i = 1; i < n; i++ )
        if ( a[ i ] > m )
            m = a[ i ];       (*)
    return m;
}
```

Här krävs att T har operationerna > och = samt kopieringskonstruktor.

Alt. 2 Returnera en kopia av det största elementet.

```
template <class T>
T max( T a[], int n ) throw ( std::range_error ) {
    if ( n < 1 )
        throw std::range_error( "max: invalid range" );
    int maxPos = 0;
    for ( int i = 1; i < n; i++ )
        if ( a[ i ] > a[ maxPos ] )
            maxPos = i;
    return A[ maxPos ];
}
```

Vid närmare eftertanke inser man att tilldelning av T-objekt vid (*) är onödig. Det räcker att hålla reda på *positionen* för det största elementet. Här räcker det att T har operationerna > och kopieringskonstruktor.

Alt. 3 Returnera en konstantreferens till det största elementet.

```
template <class T>
const T & max( T a[], int n ) throw ( std::range_error ) {
    if ( n < 1 )
        throw std::range_error( "max: invalid range" );
    int maxPos = 0;
    for ( int i = 1; i < n; i++ )
        if ( a[ i ] > a[ maxPos ] )
            maxPos = i;
    return A[ maxPos ];
}
```

Då `max` här returnerar en *referens* till T behöver T ej ha kopieringskonstruktor eftersom returvärdet ej kopieras. Mottagaren kan då välja om returvärdet skall kopieras eller ej. Detta funktionssätt är praktiskt om man t.ex. bara vill inspektera maxelementet i ett fält, t.ex. för att skriva ut det vilket typiskt hanteras på följande sätt:

```
Person pv[ 100 ];
...
const Person & pmax = max( pv, 100 );
// pmax är nu en referens till något element i pv
cout << pmax;   // ok
pmax = ...      // nok
```

Eftersom referensen `pmax` refererar till ett konstant objekt kan man inte via den ändra fältelementet. Effekten blir alltså densamma som i alt. 1, men utan kopiering. Endast > krävs således för T i denna version.

v.g.v.

Alt. 4 Returnera *positionen* för det största elementet.

```
template <class T>
int max( T a[], int n ) throw ( std::range_error ) {
    if ( n < 1 )
        throw std::range_error( "max: invalid range" );
    int maxPos = 0;
    for ( int i = 1; i < n; i++ )
        if ( a[ i ] > a[ maxPos ] )
            maxPos = i;
    return maxPos;
}
```

Den enda operationen som krävs för T är >.

Uppgift 5 (8+12 p)

a) (8 p)

Ingen lösning ges. Se inlämningsuppgiften.

b) (12 p)

```
// Naive inefficient algorithm
void processFiles( istream &textStrm, istream &substitutions ) {
    String textLine;
    while ( true ) {
        getline( textStrm, textLine );
        if ( textStrm.eof() )
            break;
        // Try every correction on this line
        String oldWord, newWord;
        while ( true ) {
           substitutions >> oldWord >> newWord;
           if ( substitutions.eof() )
              break;
           textLine.substitute( oldWord, newWord, true );
        }
        cout << textLine << endl;
        // Reset the substitution stream
        substitutions.seekg(0);
        substitutions.clear();
    }
}

void main( int argc, char *argv[] ) {
    if ( argc != 3 ) {
        cerr << "Usage: " << argv[ 0 ] << " file wordlist" << endl;
        exit( 1 );
    }
    ifstream textStream( argv[ 1 ] );
    if ( ! textStream ) {
        cerr << "Cannot open " << argv[ 1 ] << endl;
        exit( 1 );
    }
    ifstream substitutionStream( argv[ 2 ] );
    if ( ! substitutionStream) {
        cerr << "Cannot open " << argv[ 2 ] << endl;
        exit( 1 );
    }
```

```
    processFiles( textStream, substitutionStream );
}
```