Lösningsförslag till tentamen * *Preliminär*

Kursnamn	**Programmeringsteknik D, fk (3p)**
Tentamensdatum	**2001-04-21**
Program	**DAI, åk 1**
Läsår	**99/00, 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 (10 p)

```
class PasswordChecker {
public:
    PasswordChecker();
    bool check( const String password );
private:
    time_t previous;
};

PasswordChecker::PasswordChecker() : previous(time(0)) { }

bool PasswordChecker::check( const String password ) {
    if ( time(0) - previous < 1 )
        return false;
    else {
        previous = time(0);
        return lookup( password );
    }
}
```

Uppgift 3 (10 p)

Utskriften blir 1 3 3. Klassen C har en kopieringskonstruktor som kopierar grunt. I funktionen f kommer därför både x och y att vara grunda kopior av c. I alla tre objekten pekar alltså value ut samma dynamiskt allokerade heltalsvariabel. När funktionen returnerar anropas destruktorn först för y varvid heltalsvariabeln avallokeras, men när den anropas för x kraschar programmet eftersom variabeln då redan är avallokerad. (Programmet kraschade vid test med Microsoft visual C++ 6.0 men ej med Bc5.01.)

Uppgift 4 (18 p)

a) (2 p)

```
class Sentence {
...
private:
    String *theWords;
    int capacity;
    int wordCount;
};

Sentence::Sentence() : theWords(NULL),capacity(0),wordCount(0){}
Sentence::~Sentence() { makeEmpty(); }
```

b) (4 p)

```
void Sentence::addWord( const String & w ) {
    if ( capacity == 0 )
        theWords = new String[ capacity = 1 ];
    else if ( wordCount == capacity ) { // double
        String *old = theWords;
        theWords = new String[ capacity *= 2 ];
        for ( int i = 0; i < wordCount; i++ )
            theWords[ i ] = old[ i ];
        delete [] old;
    }
    theWords[ wordCount++ ] = w;
}
```

c) (1 p)

```
String & Sentence::operator[]( int i ) throw (std::out_of_range)
{
    if ( i < 0 || i >= wordCount )
        throw std::out_of_range("Sentence::operator[]");
    return theWords[ i ];
}
```

d) (3 p)

```
int Sentence::chop( String & word, String & s ) {
    int first = 0;
    // find first letter in s
    while ( first < s.Length() && ! isalnum( s[ first ] ) )
       first++;
    // find last letter in s
    int last = first;
    while ( last < s.Length() && isalnum( s[ last ] ) )
        last++;
    --last;
    // get the word
    word = s.Slice( first, last );
    // strip it off from s
    s = s.Delete( 0, last );
    return word.Length();
}
```

forts.

```
void Sentence::chopAll( const String & s ) {
    String t = s, w;
    while ( chop( w, t ) > 0 )
        addWord( w );
}
```

e) (2 p)

```
istream & operator>> ( istream & in, Sentence & s ) {
    String buf;
    char c;
    while ( in.get( c ) && c != '.' )
        buf += c;
    s.makeEmpty();
    s.chopAll( buf );
    return in;
}
```

f) (2 p)

```
void scramble( Sentence & s ) {
    for ( int i = 0; i < s.length(); i++ )
        swap( s[ random( s.length() ) ],
              s[ random( s.length() ) ] );
}
```

g) (4 p)

```
void scrambleFile( const char *fileName ) {
    ifstream in( fileName );
    if ( ! in ) {
        cerr << "Cannot open " << fileName << endl;
        exit( 1 );
    }
    Sentence s;
    while ( in >> s ) {
        scramble( s );
        cout << s << endl;
    }
    in.close();
}

void main( int argc, char *argv[] ) {
    randomize();
    if ( argc > 2 ) {
        cerr << "Usage: " << argv[ 0 ] << " file" << endl;
        exit( 1 );
    } else
        try {
            scrambleFile( argv[ 1 ] );
        }
        catch ( exception &e ) {
            cout << typeid(e).name() << ": " << e.what() << endl;
        }
}
```

Texten var förresten: "*The language has rather primitive string handling capabilities. A way to improve the situation is to build a string class. It should contain operations which cover the most common situations.*"

Uppgift 5 (12 p)

Här följer en generisk version som klarar poster av godtycklig typ. Funktionsmallen anropas

swap<*posttyp*>(*strm1*, *pos1*, *strm2*, *pos2*);

tyvärr går den inte att kompilera med Bc5.01.

```
/* N.B. It is of vital importance that the file positions are set just
before each i/o operation, otherwise wrong behaviour results if f1 and
f2 refer to the same stream.
*/

template <class T>
bool swap( fstream & f1, size_t pos1, fstream & f2, size_t pos2 ) {
    T temp1, temp2;
    // Convert to byte pos
    pos1 *= sizeof(T);
    pos2 *= sizeof(T);

    // Compute size of f1 and check limits
    f1.seekg( 0, ios::end );
    if ( pos1 < 0 || pos1 > f1.tellg() - sizeof(T) )
        return false;
    // read a record at pos1 in f1
    f1.seekg( pos1, ios::beg );
    f1.read( (char*)&temp1, sizeof(T) );

    // Compute size of f2 and check limits
    f2.seekg( 0, ios::end );
    if ( pos2 < 0 || pos2 > f2.tellg() - sizeof(T) )
        return false;
    // read a record at pos2 in f2
    f2.seekg( pos2, ios::beg );
    f2.read( (char*)&temp2, sizeof(T) );

    // rewrite the two records, but swapped
    f1.seekp( pos1, ios::beg );
    f1.write( (char*)&temp2, sizeof(T) );
    f2.seekp( pos2, ios::beg );
    f2.write( (char*)&temp1, sizeof(T) );
    return true;
}
```