CHALMERS LINDHOLMEN
Institutionen för data- och elektroteknik
Uno Holmer
Uno.Holmer@chl.chalmers.se

Lösningsförslag till tentamen * *Preliminär*

Kursnamn	**Programmeringsteknik D, fk (3p)**
Tentamensdatum	**2000-05-27**
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 (8 p)

Se kurslitteraturen.

Uppgift 2 (8+12 p)

a) (8 p)

```
String String::Delete( int From, int To ) const {
     return Replace( From, To, "" );
}
```

b) (12 p)

```
void Crack( const UserInfo & U ) {
    static const String InsertChars =
                    "0123456789!¤#%&/()=?`[]}^~*'-+_.:,;|<>";
    String Login = U.Login(),
           Pwd = U.Password(),
           Name = U.Name();
    // Is the password too short ?
    if ( Pwd.Length() < 6 )
        cout << Name << endl;

    // Is it a simple variation of the login name?
    for ( int I = -1; I < (int)Login.Length(); I ++ ) {
        // Is it altered by a one character deletion?
        String L2 = Login.Delete( I, I ); // I =-1 => L2 = Login
        if ( L2 == Pwd || L2.Reverse() == Pwd )
            cout << Name << endl;
        // Is it altered by a one character insertion?
        for ( int J = 0; J < InsertChars.Length(); J++ ) {
            for ( int K = -1; K < (int)L2.Length() + 1; K++ ) {
                String L3 = L2.Insert( K, InsertChars[ J ] );
                if ( L3 == Pwd || L3.Reverse() == Pwd )
                    cout << Name << endl;
            }
        }
    }
}
void main() {
    ifstream Password("passwd.txt");
    while ( true ) {
        UserInfo U;
        Password >> U;
        if ( Password.eof() )
            break;
        Crack( U );
    }
}
```

Uppgift 3 (8 p)

Programmet skriver ut 1 och 4. Fyra C-objekt skapas dynamiskt, men bara två återvinns, nämligen de med värdena 1 och 4. Funktion F orsakar två minnesläckor. F har två parametrar, en värdeparameter P_F, och en referensparameter Q_F. Eftersom P_F är en värdeparameter försvinner den när F terminerar och det utpekade C-objektet (3) blir en minnesläcka. Den aktuella parametern P_G ändras ej. Q_F refererar till den aktuella parametern Q_G. När Q_F tilldelas adressen till ett nytt C-objekt (4) skrivs därför Q_G över och dess tidigare utpekade C-objekt (2) tappas bort och blir en minnesläcka.

Uppgift 4 (12 p)

```
class ProgramTimer {
public:
    ProgramTimer( const char *TimeFile );
    ~ProgramTimer();
private:
    time_t StartTime, TotalTime;
    int NoOfExecutions;
    char theTimeFile[ 100 ];
};

ProgramTimer::ProgramTimer( const char *aTimeFile ) {
    strcpy( theTimeFile, aTimeFile );
    ifstream TimeStrm( theTimeFile, ios::binary );
    if ( ! TimeStrm ) {
        TotalTime = 0;
        NoOfExecutions = 0;
    } else {
        TimeStrm.read( (char*)&TotalTime, sizeof(time_t) );
        TimeStrm.read( (char*)&NoOfExecutions, sizeof(int) );
        TimeStrm.close();
    }
    StartTime = time(0);
}

ProgramTimer::~ProgramTimer() {
    ofstream TimeStrm( theTimeFile, ios::binary );
    if ( ! TimeStrm ) { // non fatal error, continue!
        cerr << "Cannot create: " << theTimeFile << endl;
        return;
    }
    TotalTime += time(0) - StartTime;
    NoOfExecutions++;
    TimeStrm.write( (char*)&TotalTime, sizeof(time_t) );
    TimeStrm.write( (char*)&NoOfExecutions, sizeof(int) );
    TimeStrm.close();
    cout << "This program has been executed ";
    cout << NoOfExecutions;
    cout << " times for a total of ";
    cout << TotalTime;
    cout << " seconds" << endl;
}
```

Uppgift 5 (12 p)

```
// Returns size of file measured in bytes
streampos GetSize( ifstream &f ) {
    f.seekg( 0, ios::end );      // Find end of stream
    streampos Size = f.tellg();  // End position
    f.seekg( 0 );                // Reset to beginning
    return Size;
}

void ReadFile( int* &Buf, streampos &NoOfInts, char *Fname ) {
    ifstream f( Fname, ios::binary );
    if ( ! f ) {
        cerr << "ReadFile: Cannot open " << Fname << endl;
        exit( EXIT_FAILURE );
    }

    streampos NoOfBytes = GetSize( f );    // How many bytes?
    NoOfInts = NoOfBytes/sizeof( int );

    // Allocate a buffer if possible
    Buf = new int[ NoOfInts ];
    if ( ! Buf ) {
        cerr << "ReadFile: Out of memory" << endl;
        exit( EXIT_FAILURE );
    }
    // Read the file into Buf in one single i/o operation
    f.read( (char *) Buf, NoOfBytes );
    if ( ! f.good() ) {
        delete [] Buf;
        Buf = NULL;
        cerr << "ReadFile: Read error" << endl;
        exit( EXIT_FAILURE );
    }
    f.close();
}

void PrintMedian( int *Buf, streampos N ) {
    if ( N == 0 )
      cout << "The data set is empty, median undefined" << endl;
    if ( N % 2 == 1 )
      cout << Buf[ N / 2 ];
    else
      cout << ( Buf[ N / 2 - 1 ] + Buf[ N / 2 ] ) / 2.0 << endl;
}

void main () {
    char FileName[100];
    cout << "File name: ";
    cin >> FileName;
    int *Buf;
    streampos N;
    ReadFile( Buf, N, FileName );
    Sort( Buf, N );
    PrintMedian( Buf, N );
}
```