TENTAMEN: Programmeringsteknik D, fk

Läs detta!

- *Uppgifterna är inte avsiktligt ordnade efter svårighetsgrad.*
- Börja varje uppgift på ett nytt blad.
- Skriv ditt namn och personnummer på varje blad (så att vi inte slarvar bort dem).
- **Skriv rent dina svar. Oläsliga svar r ä t t a s e j!**
- Programmen skall skrivas i C++, vara indenterade och kommenterade, och i övrigt vara utformade enligt de principer som lärts ut i kursen.
- Onödigt komplicerade lösningar ger poängavdrag.
- Programkod som finns i tentamenstesen behöver ej upprepas.
- Givna deklarationer, parameterlistor, etc. får ej ändras, såvida inte annat sägs i uppgiften.
- Läs igenom tentamenstesen och förbered ev. frågor.

I en uppgift som består av flera delar får du använda dig av funktioner klasser etc. från tidigare deluppgifter, även om du inte löst dessa.

Lycka till!

Uppgift 1

Välj **ett** svarsalternativ på varje fråga. Varje korrekt svar ger två poäng.

1. Vilka likheter är sanna?

```
int *P,***Q;
P = new int( 999 );
Q = new int**( &P );
```

   a. `P == Q`
   b. `P == &Q`
   c. `Q == &P`
   d. `*Q == &P`
   e. `**Q == P`
   f. `***Q == **P`
   g. `**Q == *&P`
   h. `**Q == &*P`
   i. endast a - c
   j. endast d - h
   k. högst 4 av d - h
   l. samtliga a - h
   m. inget av ovanstående

2. När anropas kopieringskonstruktorn?
   a. alltid vid initiering av nya objekt
   b. vid referensanrop
   c. vid tilldelning
   d. vid värdeanrop
   e. minst två av ovanstående
   f. inget av ovanstående

3. Om en sorteringsfunktion deklareras

```
template <class T>
void Sort( T A[], int N );
```

   vilka operationer måste vara definierade för T?
   a. `operator<` och `operator=`
   b. `operator=` och kopieringskonstruktor
   c. `operator<`
   d. `operator<` och kopieringskonstruktor
   e. `operator=`
   f. ingen av a-e, sort är dataoberoende
   g. inget av ovanstående

forts.

4. Vilket/vilka beteende är möjligt hos följande program?

```
bool G( int X ) {
    if ( X < 0 )
        throw 123;
    else
        return true;
}

int F( int X ) {
    if ( G( X ) )
        throw "F failed";
    else
        return 1;
}

void main() {
    int X;
    cin >> X;
    try {
        cout << F( X );
    }
    catch ( int Y ) {
        cout << Y << endl;
    }
}
```

a. `F failed`
b. `1`
c. `-37`
d. `123`
e. programmet avbryts
f. a och b
g. a och e
h. b och d
i. d och e
j. fler än två av a - e
k. inget av ovanstående

(8 p)

I följande uppgift används klassen String som beskrivs i bilaga i slutet av tesen.

Uppgift 2

a) Definiera medlemsfunktionen Delete i klassen String som returnerar en ny sträng med alla tecknen mellan From och To, inklusive, borttagna.

```
String Delete( int From, int To ) const;
```

(8 p)

b) Många datoranvändare utsätter sig för onödiga säkerhetsrisker genom att använda alltför simpla lösenord. Vanligen lagras användarnas lösenord, inloggningsnamn samt fullständiga namn i en speciell lösenordsfil. Filen är en textfil och kan, något förenklat, se ut som `passwd.txt` nedan [1]. Uppgiften är att skriva ett program som kan avslöja användare med lösenord som är för lätta att gissa eller räkna ut. Programmet skall undersöka om lösenorden är varianter av inloggningsnamnen och bildade efter något/några av följande mönster:

1. inloggningsnamnet med noll eller ett tecken borttaget
2. inloggningsnamnet med ett extra tecken instoppat i valfri position
3. 1 och 2 i kombination
4. omvändningen av en sträng bildad enligt något av 1-3

Dessutom underkänns alla lösenord med färre än sex tecken. Exempel:

`passwd.txt`

```
olsson:olsson:Jan Olsson
petrsson:petersson:Karl Petersson
nossnevs:svensson:Lena Svensson
nossreda:andersson:Anders Andersson
xghftrw7:smart:Lisa Smart
nilss!on:nilsson:Lisa Nilsson
no+sskre:eriksson:Allan Eriksson
xyzwv:johansson:Eva Johansson
```

Utskrift

```
Jan Olsson
Karl Petersson
Lena Svensson
Anders Andersson
Lisa Nilsson
Allan Eriksson
Eva Johansson
```

Till hjälp för att hantera information om datoranvändare finns följande klass:

```
class UserInfo {
public:
    String Password() const;     // returnerar lösenordet
    String Login() const;        // returnerar inloggningsnamnet
    String Name() const;         // returnerar hela namnet
    friend istream & operator>> ( istream & In, UserInfo & U );
private:
    ... // intern representation
};
```

Inläsningsoperatorn läser in en rad från en lösenordsfil kopplad till en inström och lagrar informationen i ett objekt av klassen U. De tre övriga funktionerna kan användas för att få tag i respektive komponent. Du kan anta att det finns en strängkonstant

```
const String InsertChars = "0123456789!¤#%&/()=?`[]}^~*'+-_.,;|<>";
```

som innehåller den teckenuppsättning användaren kan antas ha valt ett extra tecken ur för att stoppa in i lösenordet för att förvilla ev. inkräktare. All stränghantering skall göras med den bifogade strängklassen.

(12 p)

[1] I verkligheten brukar lösenorden i lösenordsfilens vänstra fält vara krypterade så att filen inte behöver hållas hemlig. Vi bortser från denna komplikation här och antar att filen kan hållas dold för insyn så att lösenorden kan lagras i klartext i filen utan att säkerheten äventyras.

Uppgift 3

Vad skriver följande program ut? Uppkommer några minnesläckor i någon av funktionerna? Motivera svaret!

```
class C {
public:
    C( int X ) : Id(X) {}
    ~C() { cout << Id; }
    int Id;
};

void F( C *P, C *&Q ) {
    P = new C( 3 );
    Q = new C( 4 );
}

void G() {
    C *P = new C( 1 );
    C *Q = new C( 2 );
    F( P, Q );
    delete P;
    delete Q;
}

void main() {
    G();
}
```

(8 p)

Uppgift 4

Många fordonstyper har en kilometerräknare som håller reda på fordonets totala körsträcka. Analogt kan det ibland vara intressant att veta hur många gånger, och hur lång sammanlagd tid, ett visst datorprogram har exekverat. Uppgiften går ut på att konstruera en klass

```
class ProgramTimer {
public:
    ProgramTimer( const char *TimeFile );
    ~ProgramTimer();
private:
    // private data
};
```

som kan användas för sådana mätningar. Man skall kunna utrusta ett program med en tidmätare enbart genom att definiera ett objekt av klassen i början av huvudprogrammet på följande sätt

```
void main() {
    ProgramTimer T("saved_timer_data.dat");
    ... resten av huvudprogrammet (som vanligt) ..
}
```

Om den allra första exekveringen av programmet tar 53 sekunder skall man få utskriften

```
This program has been executed 1 times for a total of 53 seconds
```

Om en senare exekvering ger utskriften

```
This program has been executed 123 times for a total of 8726 seconds
```

och nästa tar 14 sekunder skall utskriften då bli

```
This program has been executed 124 times for a total of 8740 seconds
```

För att kunna hålla reda på antalet exekveringar och total sammanlagd exekveringstid skall klassen spara dessa data i en binärfil. Namnet på denna ges som argument när objektet skapas. Datafilen finns ej före den första exekveringen, utan skapas då av programmet. Klassen skall ej ha några andra operationer än de som antytts ovan, dock är det tillåtet att införa dataattribut. Tid kan mätas med standardfunktionen time() som returnerar förfluten tid sedan 1971-01-01.

(12 p)

Uppgift 5

Medianen i en datamängd definieras som det storleksmässigt mittersta elementet. Det finns alltså lika många element som är mindre än medianen som större. Om datamängden innehåller ett jämnt antal element definieras medianen som medelvärdet av de två mittersta.
Exempel: Medianen av heltalen

1 22 10 3 97 4 2

är 4, medan 3 2 1 4 har medianen 2.5. Det är lätt att beräkna medianen i en sorterad datamängd. Skriv ett program som beräknar medianen för en datamängd bestående av heltal som är lagrade i en binärfil. Filen läses lämpligen först in till ett fält. Följande färdiga funktion för att sortera heltalsfält är given:

```
void Sort( int Tab[], int N );
```

Parametern N anger hur många tal i Tab som skall sorteras. Programmet skall fungera för filer av i princip godtycklig storlek. Det minne som allokeras för fältet skall motsvara filens storlek. En övre begränsning av antalet element får ej antas.

(12 p)

Bilagor till tentamen

Strängklassens operationer

***Definition**: I operationerna Slice, Replace, och Delete betraktas ett intervall som tomt om S.Length() ≤ From ≤ To eller From ≤ To < 0 eller From > To.*

S = T	Tilldelar T till S
S = A	Tilldelar teckenfältet A till strängen S. A skall vara avslutad med ett '\0'-tecken.
S = c	Tilldelar c till S.
S += c	Lägg till c sist i S.
S[I]	Indexering med indexkontroll. Alla strängar indexeras med början på 0. Vid indexfel kastas ett undantag.
S.Slice(From, To)	Ger en ny sträng som innehåller tecknen från och med From och till och med To i S. Om From < 0 eller To ≥ S.Length() justeras de till början resp. slutet. Ligger båda utanför blir t.ex. resultatet S.Slice(0,S.Length()-1). Om intervallet är tomt returneras en tom sträng.
S.Insert(I, T)	Ger en ny sträng med T insatt efter position I. Om I < 0 sätts T in före övriga tecken resp. efter om I ≥ S.Length().
S.Replace(From, To, T)	Ger en ny sträng med tecknen från och med From och till och med To ersatta med T. Om From eller To ligger utanför strängens gränser justeras dessa till början resp. slutet, analogt med Slice. Om intervallet är tomt returneras S.
S.Delete(From, To)	Ger en ny sträng där tecknen från och med From och till och med To är borttagna. Om From eller To ligger utanför strängens gränser justeras dessa till början resp. slutet, analogt med Slice. Om intervallet är tomt returneras S.
S + T	Ger en ny sträng som är sammansättningen av S och T.
S + c	Ger en ny sträng som är sammansättningen av S och c.
c + S	Ger en ny sträng som är sammansättningen av c och S.
S.Match(From, T)	Ger första positionen från och med From där alla tecknen i T matchar (är lika med) tecknen i S. Om ingen sådan position finns returneras –1.
S.Reverse()	Ger en ny sträng som är omvändningen av S.
S.ToArray(A)	Kopierar tecknen i S till fältet A. A måste vid anropet vara tillräckligt stort för att rymma tecknen i S, annars är resultatet odefinierat vilket innebär att användaren av klassen har ansvaret för följderna, inte klasskonstruktören.
S.Length()	Ger antalet tecken i S.
S == T	Sant om S och T är lika.
S < T	Sant om S kommer före T i bokstavsordning (lexikografisk ordning, ASCII).
Ström >> S	Läser in ett ord till S från Ström (samma beteende som >> för teckenfält).
Ström << S	Skriver ut S på Ström.
S.GetLine(Ström)	Läser in en hel textrad från Ström till S. Nyradstecknet lagras ej i S.

```
class String {
public:
    // Constructors
    String( );
    String ( char C );
    String( const char * Value );
    String( const String & Value );

    // Destructor
    ~String( );

    // Assignment operators
    const String & operator=( const String & Rhs );
    const String & operator=( const char * Rhs );
    const String & operator+= ( char C );

    // Selection and insertion
    char operator[ ]( int Index ) const;  // Rvalue
    char & operator[ ]( int Index );      // Lvalue
    String Slice( int From, int To ) const;
    String Insert( int After, const String  & Value ) const;
    String Replace( int From, int To, const String  & Value ) const;
    String Delete( int From, int To ) const;

    // Concatenation operators
    String operator+ ( char C ) const;
    friend String operator+ ( char C, const String & Rhs );
    String operator+ ( const String & Rhs ) const;

    // Miscelaneous operators
    int Match( int Start, const String & Pattern ) const;
    String Reverse() const;
    char *ToArray( char *Buf ) const;
    unsigned int Length( ) const;

    // Friends for comparison
    friend bool operator== ( const String & Lhs, const String & Rhs );
    friend bool operator< ( const String & Lhs, const String & Rhs );

    // I/IO
    friend istream & operator>>( istream & In, String & Value );
    friend ostream & operator<<( ostream & Out, const String & Value );
    void GetLine( istream & In );
private:
     // Data representation
};
```