# 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, garderingar ger noll poäng. Motiveringar krävs ej.

1. Vilka påståenden om P och Q är korrekta?

```
int *P,**Q;
P = new int( 123 );
Q = new int*( P );
```

   a. P och Q är lika
   b. &P och Q är lika
   c. &P och Q är samma
   d. &P och *Q är lika
   e. &P och *Q är samma
   f. P och *Q är lika
   g. P och *Q är samma
   h. *P och **Q är lika
   i. *P och **Q är samma
   j. b och c
   k. d och e
   l. b och e
   m. f, h och i
   n. f, g och h
   o. alla utom a
   p. inget av ovanstående

2. Vilka påståenden stämmer för text- och binärfiler?
   a. i en binärfil lagras radslutstecknet i ett 32-bitars ord
   b. textfiler kan endast bearbetas sekventiellt, ej direkt
   c. både text- och binärfiler kan bearbetas med formatterad in- och utmatning
   d. om man glömmer att stänga en fil är den fortfarande öppen vid nästa exekvering
   e. fler än ett av ovanstående
   f. inget av ovanstående

3. Om funktionen Min definieras

```
template <class T>
const T & Min( const T & x, const T & y ) {
    if ( x < y )
        return x;
    else
        return y;
}
```

   vilka operationer måste vara definierade för T?
   a. operator< och operator=
   b. `operator=` och kopieringskonstruktor
   c. operator< enbart
   d. operator< och kopieringskonstruktor
   e. inget av ovanstående är korrekt

*forts.*

4. Ett tillräckligt villkor för att de två C-objekten skall deallokeras är:

```
Stack<C*> *S = new Stack<C*>;
S->Push( new C );
S->Push( new C );
delete S;
```

a. klassen C har en destruktor
b. stackklassen har en destruktor
c. båda klasserna har destruktorer
d. det räcker inte att båda klasserna har destruktorer

5. Vilket/vilka beteende är omöjligt om man ger talet 5 som indata (`random` (X)∈ [0,X-1])?

```
int G( unsigned int X, unsigned int Y ) {
    if ( Y == 0 ) throw -1; else return X/Y;
}

int F ( int X ) {
    try {
        int Z = G( X, random( X ) );
        if ( Z < 3 )
            throw "F failed";
        else
            return 2*Z;
    }
    catch ( int A ) {
        return -A;
    }
}

void main() {
    randomize();
    int X;
    cin >> X;
    try {
        cout << F( X ) << endl;
    }
    catch ( const char *Msg ) {
        cout << Msg << endl;
    }
}
```

a. `F failed`
b. `-3`
c. `10`
d. `8`
e. `1`
f. två av ovanstående
g. inget av ovanstående

(10 p)

### Uppgift 2

I säkerhetssystemet i ett visst operativsystem finns en klass som sköter kontrollen av användarnas lösenord vid inloggning:

```
class PasswordChecker {
public:
    ...
    bool check( const String password );
private:
    // data representation
};
```

Funktionen `check` kontrollerar med hjälp av funktionen `lookup` i en databas om ett angivet lösenord är korrekt. För att öka säkerheten mot upprepade intrångsförsök gör `check` högst en kontroll mot databasen per sekund. Anropas `check` oftare returnerar den `false`. Komplettera klassen med nödvändiga dataattribut och ev. ytterligare funktioner. Implementera samtliga medlemsfunktioner. Standardfunktionen `time(0)` returnerar förfluten sekundtid sedan 1971-01-01. Funktionen `lookup` är given och har prototypen:

```
bool lookup( const String & password );
```

den returnerar `true` om det angivna lösenordet finns i databasen och `false` annars. Anropet av `lookup` kan antas ta försumbar tid.

(10 p)

### Uppgift 3

Vad händer troligen vid exekvering av följande program?

```
class C {
public:
   C( int x ) { value = new int(x); }
   C( const C & rhs ) { value = rhs.value; (*value)++; }
   ~C() { delete value; }
   void info() { cout << *value << endl; }
private:
   int *value;
};

void f( C x ) { C y = x; x.info(); y.info(); }

void main() {
   C c(1);
   c.info();
   f( c );
   c.info();
}
```

Motivera svaret!

(10 p)

## Uppgift 4

Söndagsbilagan i den engelsksprakiga tidningen Lindholmen News har en pysselsida med diverse knep och knap. Ett av problemen brukar vara att flytta om orden i en slumpmässig text så att innehållet blir begripligt. Redaktionen behöver ett datorprogram för att skapa sådana ordpussel. Programmet skall givet textfiler med "riktig text" skriva ut texten med orden slumpmässigt omkastade inom varje mening. Filerna antas för enkelhets skull innehålla meningar som alltid avslutas med '.' (punkt). Orden består av alfanumeriska tecken. Meningarna kan sträcka sig över flera textrader. En körning med programmet kan t.ex. se ut så här: (*string.txt innehåller tre meningar ur ett stycke text - om vad?* ☺)

```
C:\program\LindhNews\SundayPuzzle>makepuzzle string.txt
rather has string language The handling primitive capabilities.
the way a A improve to is to string situation build class.
operations contain which most the cover It should common situations.
```

För att hantera meningar finns följande påbörjade klass.

```
// A sentence is an ordered collection of words.
class Sentence {
public:
      // Create an empty sentence
    Sentence();
    ~Sentence();
      // Return the number of words in the sentence
    int length() const;
      // Empty the sentence
    void makeEmpty();
      // Add a word at the end of the sentence
    void addWord( const String & word );
      // Access the i:th word in the sentence (L and R value)
    String & operator[]( int i ) throw (std::out_of_range);
      // Read a sentence up to a '.'
    friend istream & operator>> ( istream & in, Sentence & s );
      // Write a sentence. Add a '.' at the end.
    friend ostream & operator<< ( ostream & out,
                                  const Sentence & s );
private:
      // Extract the first word from s into w,
      // and set s to the remaining characters.
      // The number of chars in w is returned.
    int chop( String & w, String & s );
      // Iteratively chop all the words in s
      // and store them in the sentence.
    void chopAll( const String & s );

    // You have to define a data representation!
};


// utility function template
template <class T>
void swap( T & x, T & y ) {
    T temp = x;
    x = y;
    y = temp;
}
```

I a-f krävs att du väljer en lämplig datarepresentation för objekt av klassen `Sentence`. Datarepresentationen skall bygga på dynamisk minneshantering enligt de principer som lärts ut i kursen. Ett objekt av klassen `Sentence` skall i princip kunna innehålla hur många ord som helst. Klassen skall själv sköta allokering av mer minne vid behov. All stränghantering skall baseras på klassen `String` (se bilaga).

a) Implementera konstruktor och destruktor.

(2 p)

b) Implementera funktionen `addWord` som lägger till ett ord sist i ett meningsobjekt.

(4 p)

c) Implementera `operator[]`. Angivet undantag med lämplig feltext skall kastas för ogiltigt index.

(1 p)

d) Implementera hjälpfunktionerna `chop` och `chopAll`. Funktionen `chop` skall leta upp första ordet i `s`. Ordet returneras i `w` och tas dessutom bort från `s`. Dessutom skall funktionen returnera längden på det funna ordet. Ord är uppbyggda av alfanumeriska tecken, vilket kan testas med standardfunktionen `isalnum`. Funktionen `chopAll` lägger till alla ord som finns i `s` till ett meningsobjekt.

(3 p)

e) Implementera `operator>>`. Funktionen skall läsa in en mening fram till '.' eller filslut, beroende på vilket som inträffar först, till ett meningsobjekt. Efter operationen skall objektet innehålla alla orden i meningen.

(2 p)

f) Implementera den fristående funktionen `scramble`. Funktionen skall kasta om orden i en mening slumpmässigt. Standardfunktionen `random`(N) ger ett slumptal i intervallet [0,N).

(2 p)

g) Skriv ett program som givet en textfil med meningar som indata skriver ut dem på skärmen med orden slumpmässigt omkastade inom varje mening. Programmet skall kunna köras i DOS och ta filnamnet som parameter, som exemplet ovan visar. Fel vid filöppning skall hanteras på lämpligt sätt. Eventuella standardundantag skall hanteras med utskrift av både undantagsnamn och ingående feltext. Dela in i lämpliga funktioner.

(4 p)

## Uppgift 5

En binärfil har poster av följande schematiska format:

```
struct P {
    Typ1 v1;
    ...
    Typn vn;
};
```

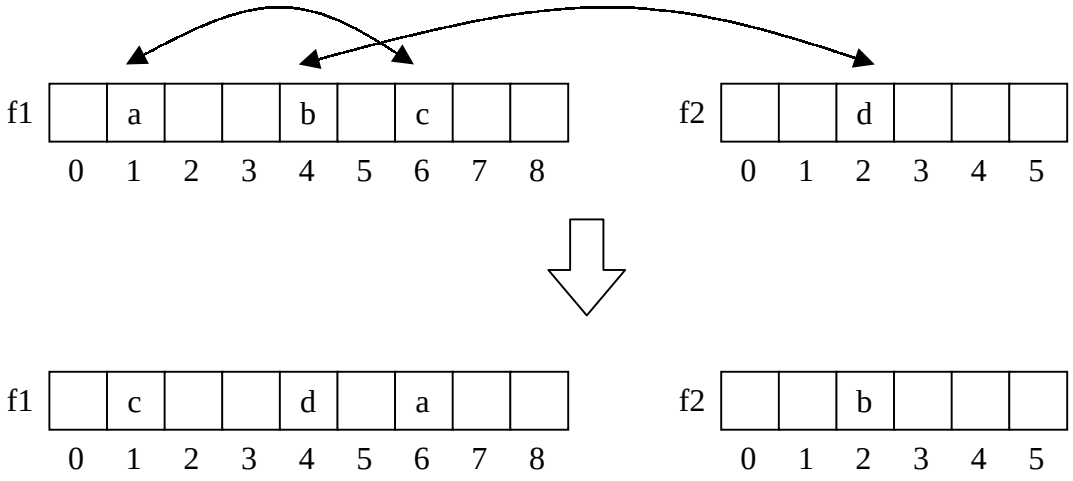
Posterna i en binärfil med N poster kan, analogt med vektorer, numreras 0..N-1 och indexeras med direktaccessteknik. Ibland finns behov av att låta två poster i en eller flera filer byta plats med varandra. Det vore då bekvämt att kunna ange posternas index istället för komplicerade sökoperationer. Implementera funktionen

```
bool swap( fstream & f1, size_t pos1, fstream & f2, size_t pos2 );
```

som byter innehåll på platserna `pos1` och `pos2` i filerna som `f1` och `f2` är kopplade till. Typen size_t används för att ange filpositioner. Antag att båda filerna är binärfiler och enbart innehåller poster av typen P.[1] Funktionen skall kontrollera att de angivna positionerna anger poster som finns i respektive fil. Om så är fallet byts posterna i `pos1` och `pos2` mot varandra och funktionen returnerar `true`, annars ändras ingen av filerna och funktionen returnerar `false`. Antag att båda filerna är öppna vid anropet. Funktionen skall även hantera fallet när `f1` och `f2` är kopplade till samma fil. Exempel: Operationerna

```
swap( f1, 1, f1, 6 );
swap( f1, 4, f2, 2 );
```

förändrar `f1` och `f2` enligt följande:




(12 p)

[1] Funktionen kan generaliseras genom att göra posttypen till en typparameter, men det krävs ej i denna uppgift.

# Bilagor till tentamen

***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
};
```