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. Vilket påstående är korrekt om egna klasser
   a. inga operatorer är automatiskt definierade
   b. alla operatorer är automatiskt definierade
   c. tilldelningsoperatorn är aldrig automatiskt definierad
   d. överlagrade operatorers prioritet kan ändras
   e. inget av ovanstående

2. Vilket uttryck är korrekt om klassen C har en medlem x och p har typen pekare till C?
   a. `p.x`
   b. `*p.x`
   c. `p->x`
   d. `(*p)->x`
   e. `C.x`
   f. `C->x`
   g. inget av ovanstående

3. Vad händer vid exekvering av

```
class A {
public:
    A( int *x ) { p = new int*(x); }
    ~A() { delete p; }
private:
    int **p;
};

class B {
public:
    B( int x ) { q = new int(x); ap = new A(q); }
    ~B() { delete ap; delete q; }
private:
    A *ap;
    int *q;
};

void main() {
    B *b = new B(123);
    delete b;
}
```

   a. allt minne utpekat av `b` frigörs
   b. en minnesläcka uppstår
   c. programmet kraschar
   d. inget av ovanstående

forts.

4. Vad skrivs ut?

```
int *a[] = { new int( 1 ), new int( 2 ), new int ( 3 ), NULL };
int **p = a;
*++p = *(p + 1);
++**p;
for ( int **q = a; *q; q++ )
   cout << **q;
```

a. 1 2 3
b. 1 2 4
c. 1 3 3
d. 1 3 4
e. 1 4 4
f. 2 2 3
g. inget av ovanstående

5. Studera följande kodavsnitt:

```
int x, y;
cout << "x: ";
cin >> x;
do {
     strm.read( reinterpret_cast<char *>( &y ), sizeof(int) );
     if ( strm.eof() )
          break;
} while ( x != y );
x++;
```

Vilket av följande ökar första förekomsten i strömmen av det från tangentbordet inlästa talet med 1 på korrekt sätt?

```
// a
strm.seekp( -sizeof(int), ios::cur );
strm.write( reinterpret_cast<const char *>( &x ), sizeof(int) );

// b
strm.seekp( strm.tellg() - sizeof(int) );
strm.write( reinterpret_cast<const char *>( &x ), sizeof(int) );

// c
strm.write( reinterpret_cast<const char *>( &x ), sizeof(int) );

// d
strm.seekp( strm.tellg() - sizeof(int) );
strm.write( reinterpret_cast<const char *>( x ), sizeof(int) );

// e
strm.seekp( strm.tellg() - 1 );
strm.write( reinterpret_cast<const char *>( &x ), sizeof(int) );
```

f. inget av ovanstående

(10 p)

Uppgift 2

N-sidiga tärningar (*eng. Die*) kan modelleras med objekt av följande klass:

```
class Die {
public:
    Die( int Sides );
    void toss();
    int getValue() const;
private:
    // data representation
};
```

För att skapa ett tärningsobjekt måste man ange hur många sidor den har. Tärningen kastas med `toss` varvid den får ett nytt slumpmässigt värde. Värdet kan inspekteras med `getValue`. Ibland vill man hantera flera tärningar på en gång. Klassen CupOfDice kan användas för att skapa en "mugg av tärningar".

```
class CupOfDice {
public:
    CupOfDice();
    ~CupOfDice();
    CupOfDice & operator+=( Die *d );
    void shake();
    int getSum() const;
private:
    // data representation
};
```

Man kan associera tärningar till en mugg med `+=`, skaka om muggen med `shake` samt få reda på tärningssumman med `getSum`. Med associera menas att operationen `+=` inte skapar egna kopior av tärningarna som adderas utan hanterar dem via pekare.

a) Definiera en lämplig datarepresentation och implementera klassens medlemsfunktioner. Man skall kunna addera obegränsat antal tärningar till ett muggobjekt. All minneshantering skall skötas internt i klassen. Inga minnesläckor får orsakas av klassen. Destruktorn skall frigöra allt minne som allokerats internt för muggobjektet, samt även de associerade tärningarna. Du får anta att samma tärning inte har lagts till i två separata muggobjekt samt att alla tärningsobjekt som adderas till en mugg skapats dynamiskt. Klassen `Die` är given och skall ej implementeras.

(9 p)

b) Skriv ett program som skapar en tärningsmugg med några tärningar med olika sidantal samt räknar ut den genomsnittliga summan för N kast där N matas in av användaren.

(4 p)

Uppgift 3

Vilka utskrifter kan fås vid exekvering av följande program?

```
int F() {
    switch ( random( 4 ) ) {
        case 0: throw false;
        case 1: throw 123;
        case 2: throw "Positive";
        case 3: return random(10) - 5;
    }
}

int G() {
    try { if ( F() < 0 ) throw "Negative"; }
    catch ( int x ) { return 2*x; }
    catch ( char *y ) { throw 456; }
    return 999;
}

void main() {
    randomize();
    try { cout << "G returned " << G() << endl; }
    catch ( int a ) { cout << a << endl; }
    catch ( char *b ) { cout << b << endl; }
    catch ( ... ) { cout << "Some error" << endl; }
}
```

`random`(N) returnerar ett slumptal i intervallet [0,N). Motivera svaret!

(7 p)

Uppgift 4

Funktionen

```
int max( int a[], int n );
```

returnerar det största elementet i heltalsfältet a. Parametern n anger antalet element som skall behandlas och antas vara större än noll. Ge en fullständig definition av en motsvarande generisk funktion `genericMax` som kan användas för fält av fler typer än **`int`**. Istället för att anta att n > 0 skall funktionen kasta undantaget `std::range_error` om n < 1. Dessutom skall undantaget specificeras på lämpligt sätt i funktionshuvudet. Besvara följande fråga: Vilka operationer måste vara definierade för fältelementen?

(10 p)

Uppgift 5

a) Implementera funktionen `substitute` i klassen `String` (se bilagan). *Tips:* utnyttja övriga funktioner i klassen.

(8 p)

b) En *substitutionsfil* innehåller *substitutioner*. En substitution är ett ordpar där det första ordet anger ett ord som skall bytas ut, och det andra vilket ord det skall bytas mot. I denna uppgift är substitutionsfilen en textfil med ett substitutionspar per rad. Skriv ett program som givet en textfil och en substitutionsfil byter ut alla teckenföljder i textfilen som förekommer som första komponent i något av substitutionsfilens substitutionspar. Substitutionsfilen skall ej ändras. Utskriften skall göras till skärmen. För poäng krävs att klassen `String` används för all stränghantering. Dela in programmet i lämpliga funktioner. Antag att alla substitutionerna i substitutionsfilen har unika förstakomponenter. Exempel:

före
```
This is a little
poem about the
rights of the
constitution.
```

efter
```
Thiz iz a serious
speech about the
heights of the
substitution.
```

substitutionsfilen
```
con sub
is us
song rap
us iz
poem speech
right height
little serious
```

(12 p)

Bilaga

Klassen String

I tabellen betecknar variablerna s, t och u strängobjekt, pos, len heltal, c tecken, p pekare till char, b bool och strm en textström. Muterande operationer är markerade med ett 'm' i den andra kolumnen, övriga är konstanta och ändrar ej objekten. Konstanten String::npos används för att ange en logisk position bortom varje i praktiken förekommande stränglängd. T.ex. returnerar operationen find positionen för en hittad förekomst av ett sökt tecken, men String::npos om tecknet ej hittas.

***Definition**: I operator() (delsträng) och replace betraktas intervallet [pos,pos+len) som* tomt *om s.length()≤ pos eller pos + len ≤ 0 eller len ≤ 0.*

s = t	m	Tilldelar t till s
s += c	m	Lägg till c sist i s.
s += t	m	Lägg till t sist i s.
s[pos] [+]	m	Indexering med indexkontroll. Första elementet har index 0. Om pos ej ligger i intervallet [0,s.length()-1] kastas undantaget std::out_of_range.
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 - c		Ger en kopia av s med alla förekomster av c borttagna.
s - t		Ger en kopia av s med alla förekomster av tecken som finns i t borttagna.
s(pos, len)		Ger en delsträng som innehåller högst len tecken från pos och framåt i s. Är s tom eller intervallet [pos,pos+len) tomt returneras en tom sträng. Om pos < 0 eller pos + len > s.length() justeras pos och len så att intervallet hamnar inom strängens gränser. Om t.ex. s.length() == 10 tolkas s(-2, 5) som s(0, 3), s(6, 8) som s(6, 4) och s(-2, 15) som s(0, 10).
s.insert(pos, t)	m	Sätter in t före position pos i s. Om s är tom blir s en kopia av t. Om t är tom ändras ej s. Om pos ≤ 0 sätts t in först. Om pos ≥ s.length() sätts t in sist. Om s och t är samma objekt, d.v.s. man försöker sätta in en sträng i sig själv, kastas undantaget std::invalid_argument.
s.replace(pos, len, t)	m	Ersätter upp till len tecken från pos och framåt i s med t. Är s tom eller intervallet [pos,pos+len) tomt ändras ej s. Om pos < 0 eller pos + len > s.length() justeras pos och len så att intervallet hamnar inom strängens gränser på samma sätt som för delsträngsoperatorn ovan. Om s och t är samma objekt kastas undantaget std::invalid_argument.
s.find(pos, c)		Ger första positionen från och med pos som innehåller c. Om inget sådant index finns returneras String::npos.
s.find(pos, t)		Ger första positionen från och med pos där alla tecknen i t matchar tecknen i s, d.v.s. det minsta index i ≥ pos s.a. s(i, t.length()) == t. Om inget sådant index finns returneras String::npos.
s.substitute(t, u, b)	m	Om b är falsk byts första förekomsten av t i s mot u, om b är sann byts alla ej överlappande förekomster ut. Om s och u är samma objekt kastas undantaget std::invalid_argument.
s.reverse()	m	Vänder innehållet i s baklänges.
s.copy(p, len)		Kopierar upp till len tecken från s till ett teckenfält som pekas ut av p, fältet måste vid anropet vara tillräckligt stort för att rymma de kopierade tecknen, annars är resultatet odefinierat. Det är inte ett krav att copy nullterminerar den kopierade texten om len ≤ s.length().
s.length()		Ger antalet tecken i s.
s == t		Avgör om s och t är lika.
s != t		Avgör om s och t är olika.
s < t		Avgör om s kommer före t i bokstavsordning. [*]
s > t		Avgör om s kommer efter t i bokstavsordning. [*]
s ≤ t		Avgör om s kommer före t i bokstavsordning eller s och t är lika. [*]

s ≥ t		Avgör om s kommer efter t i bokstavsordning eller s och t är lika.. *
strm >> s	m	Läser in ett sammanhängande ord till s från strm (samma beteende som >> för teckenfält).
strm << s		Skriver s till strm.
getline(strm, s)	m	Läser in en hel textrad från strm till s. Nyradstecknet lagras ej i s.

Noter till tabellen
+ Det finns även en ickemuterande variant för konstanta strängobjekt.
* Lexikografisk ordning m.a.p. ASCII-tecken avses.

```
class String {
public:
    // Constructors
    String( );
    String( const char *value );
    String( const String & value );  // Copy constructor
    // Destructor
    ~String( );

    // Assignment operators
    const String & operator=( const String & rhs );
    String & operator+= ( char c );            // single character addition
    String & operator+=( const String & rhs ); // concatenation

    // Checked indexed access
    // (L)Rvalue for non const strings
    char & operator[ ]( int index ) throw (std::out_of_range);
    // Rvalue for const Strings
    char operator[ ]( int index ) const throw (std::out_of_range);

    // Concatenation operators
    String operator+ ( char c ) const;
    friend String operator+ ( char c, const String & rhs );
    String operator+ ( const String & rhs ) const;

    // Character deletion
    String operator- ( char c ) const;
    String operator- ( const String & rhs ) const;

    // Substring selection
    String operator() ( int pos, int len ) const;

    // String insertion
    String & insert( int pos, const String & value )
     throw (std::invalid_argument);

    // String replacement
    String & replace( int pos, int len, const String & value )
          throw (std::invalid_argument);

    // Matching and substitution
    int find( int pos, char c ) const;
    int find( int pos, const String & pattern ) const;
```

forts.

```
    String & substitute( const String &oldStr,
                         const String &newStr, bool global )
          throw (std::invalid_argument);

    // Reverse contents
    String & reverse( );

    // Copy to character array
    void copy( char *buf, int len ) const;

    int length( ) const;

    // Friends for comparison
    friend bool operator== ( String & lhs, String & rhs );
    friend bool operator!= ( String & lhs, String & rhs );
    friend bool operator<  ( String & lhs, String & rhs );
    friend bool operator>  ( String & lhs, String & rhs );
    friend bool operator<= ( String & lhs, String & rhs );
    friend bool operator>= ( String & lhs, String & rhs );

    // Word input
    friend istream & operator>>( istream & in, String & value );
    // Line input
    friend void getline( istream & in, String & value );
    // Output
    friend ostream & operator<<( ostream & out, const String & value );

    static const int npos;    // a position beyond the length of any string
private:
    // data representation
};
```