TENTAMEN: Objektorienterad programutveckling

Läs detta!

- *Uppgifterna är inte 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!**
- Programkod som finns i tentamenstesen behöver ej upprepas.
- Programkod skall skrivas i C++ och vara indenterad och kommenterad.
- Onödigt komplicerade lösningar ger poängavdrag.
- Givna deklarationer, parameterlistor etc. får ej ändras.
- 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** alternativ för varje fråga! Garderingar ger noll poäng. Inga motiveringar krävs. Varje korrekt svar ger två poäng.

1. Vilken/vilka av följande aktiviteter har en framträdande roll vid OOA?
   a. indelning av klasser i moduler
   b. klassificering av objekt
   c. konstruktion av algoritmer
   d. beskrivning av objektrelationer
   e. a och b
   f. a och c
   g. b och c
   h. b och d
   i. fler än två av a-d
   j. inget av ovanstående är korrekt

2. Regressionstest innebär att
   a. testa hur delsystem fungerar ihop
   b. upprepa test efter förändringar
   c. testa delsystem separat
   d. köra testsviten baklänges
   e. inget av ovanstående är korrekt

3. Vilken relation är vanligen dubbelriktad?
   a. arv
   b. association
   c. aggregat
   d. fler än en
   e. inget av ovanstående

4. Förkortningen CORBA betecknar
   a. en objektorienterad systemutvecklingsprocess
   b. en äldre grafisk notation för klassdiagram
   c. en standard för användargränssnitt
   d. Corporative Objects Revision Buisiness Analysis
   e. inget av ovanstående är korrekt

5. Vilken typ av arv i C++ motsvarar en "är"-relation?
   a. publikt arv
   b. skyddat arv
   c. privat arv
   d. inget av ovanstående

(10 p)

Uppgift 2

a) Vad skriver följande program ut? Motivera svaret!

```
class Base {
public:
    virtual void F() { cout << "Base::F "; G(); }
    virtual void G() { cout << "Base::G "; H(); }
    virtual void H() { cout << "Base::H " << endl; }
};

class Sub1 : public Base {
public:
    void G() { cout << "Sub1::G "; H(); }
};

class Sub2 : public Sub1 {
public:
    void F() { cout << "Sub2::F "; G(); }
    void H() { cout << "Sub2::H " << endl; }
};

void main() {
    Base *A[] = { new Base, new Sub1, new Sub2, NULL };
    for ( Base **p = A; *p; p++ )
        (*p)->F();

}
```

(6 p)

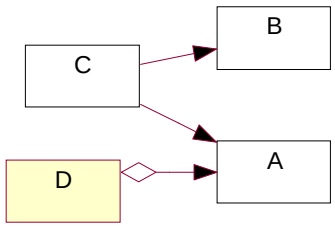
forts.

2 b)
Studera följande klassdefinitioner:

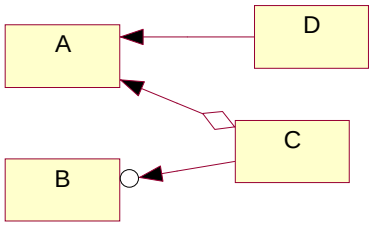
```
class A { ... };
class B { ... };
class D { ... };

class C {
public:
   C( B &Bref ) : theB(Bref) { theA = new A; }
   ~C() { delete theA; }
   // ... + övriga operationer
private:
   A *theA;
   B &theB;
};
```

Vilket av klassdiagrammen beskriver bäst möjliga inbördes relationer mellan A, B, C och D? Motivera svaret!

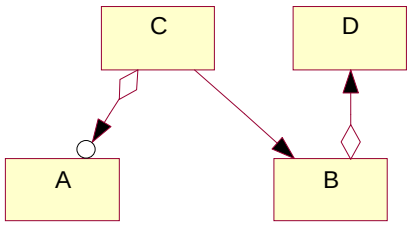
a)



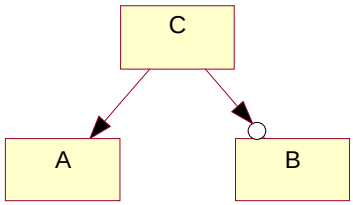
b)



c)



d)



(4 p)

Uppgift 3

Klassen `Stack` nedan saknar en operation för att vända stackens innehåll baklänges så att toppelementet hamnar i botten, och vice versa. Definiera klassen `ExtendedStack` som adderar operationen `Reverse`. Den nya klassen skall definieras genom utvidgning med arv från `Stack`.

```
// Stack class interface
//
// Etype: must have zero-parameter and copy constructor
// CONSTRUCTION: with (a) no initializer;
//    copy construction of Stack objects is DISALLOWED
// Deep copy is supported
//
// ******************PUBLIC OPERATIONS*********************
// void Push( Etype X )   --> Insert X
// void Pop( )            --> Remove most recently inserted item
// Etype Top( )           --> Return most recently inserted item
// bool IsEmpty( )        --> Return true if empty; false ow
template <class Etype>
class Stack{
public:
    Stack( );
    ~Stack( );
    const Stack & operator= ( const Stack & Rhs );
    void Push( const Etype & X );
    void Pop( );
    const Etype & Top( ) const;
    bool IsEmpty( ) const;
private:
   // private data representation
};
```

Exempel

```
ExtendedStack<int> S;
S.Push( 1 );
S.Push( 2 );
S.Push( 3 );
S.Reverse();
S.Pop();
S.Push( 4 );
S.Reverse();
while ( ! S.IsEmpty() ) {          utskrift: 3 2 4
    cout << S.Top();
    S.Pop();
}
```

(10 p)

Uppgift 4

Många spel har liknande struktur. T.ex. finns spel med två deltagare som spelar mot varandra genom att göra var sitt drag tills spelet är avgjort med vinst för någon av parterna, eller remi (oavgjort). Klassen `Game` är en basklass för klasser som implementerar spel av denna typ där en användare spelar mot datorn som motpart.

```
class Game {
public:
   void Run();
   virtual void Reset() { Status = Undecided; }
   virtual void UserMove() = 0;
   virtual void ComputerMove() = 0;
protected:
    enum GameStatus { UserWin, ComputerWin, Draw, Undecided };
    GameStatus Status;
};
```

Medlemsfunktionerna:

`Run`	kör en spelomgång
`Reset`	återställer spelstatus
`UserMove`	låter användaren göra ett drag
`ComputerMove`	gör ett drag för datorns räkning

En spelomgång kan vara i olika tillstånd:

`UserWin`	spelutgången är avgjord, användaren vann
`ComputerWin`	spelutgången är avgjord, datorn vann
`Draw`	spelutgången blev oavgjord (remi)
`Undecided`	spelutgången är ännu ej avgjord

a) Implementera klassen Game. Funktionen `Run` skall låta användaren börja. Lämpliga meddelanden skall ges beroende på spelets utgång.

(8 p)

b) Implementera en subklass `Sticks` till `Game` som låter användaren spela "tändsticksspelet" mot datorn. Spelet brukar gå till så att 21 stickor läggs upp på ett bord. Spelarna tar omväxlande bort en eller två stickor och den som tar den sista stickan förlorar. (Detta spel kan ej sluta oavgjort.). Klassens funktioner skall ge lämpliga meddelanden så att användaren löpande informeras om spelets status. Exempel. Utskrifternas exakta utseende är mindre viktigt.

```
There are 21 sticks left
You can take at most 2 sticks: 1
The computer takes 1 stick
There are 19 sticks left
You can take at most 2 sticks: 2
The computer takes 1 stick
There are 16 sticks left
You can take at most 2 sticks: 1
The computer takes 2 sticks
There are 13 sticks left
You can take at most 2 sticks: 1
The computer takes 2 sticks
There are 10 sticks left
You can take at most 2 sticks: 2
...
The computer takes 1 stick
There are 7 sticks left
You can take at most 2 sticks: 2
The computer takes 1 stick
There are 4 sticks left
You can take at most 2 sticks: 1
The computer takes 2 sticks
There are 1 sticks left
You can take at most 1 stick: 1
The computer wins
```

(8 p)

Uppgift 5

I en stugby finns stugor av olika slag att hyra, och till olika dagspriser beroende på utrustning och veckodag. Stugorna delas in i tre grupper enligt följande:

- Gröna stugor har alltid grundhyra (mellanklass).
- Gula stugor har grundhyra alla dagar utom lör-sön, då de kostar 20% mer (de luxe).
- Röda stugor har mån-fre grundhyra - 100 kr samt ytterligare 30% rabatt, övriga veckodagar grundhyra (budgetmodell).

Följande basklass är definierad för stugor:

```
class Stuga {
public:
    Stuga(long Nr, float Gpris) : _StugNr(Nr), Grundpris(Gpris) {}
    long StugNr() { return _StugNr; }
    virtual float Dagspris() = 0;
private:
    long _StugNr;
protected:
    float Grundpris;
};
```

a) Definiera och implementera en av klasserna Grön, Gul och Röd genom arv på lämpligt sätt. För att ta reda på aktuell veckodag kan du anta att följande finns givet:

```
enum Veckodag { Mån, Tis, Ons, Tor, Fre, Lör, Sön };
Veckodag Dag();  // returnerar aktuell veckodag
```

(3 p)

b) Implementera en fristående funktion Intäkt() som tar en lista av uthyrda stugor som argument och returnerar den totala hyresintäkten den aktuella dagen. För att bli godkänd krävs att datastrukturen lista används (se bilaga), samt att lösningen baseras på polymorfism.

(5 p)

c) Skriv ett huvudprogram som exemplifierar hur klasserna i a) kan användas genom att definiera en liten ”stugby” samt demonstrera hur funktionen Intäkt() kan användas för att skriva ut dagsintäkten för byn.

(3 p)

d) Ett mindre antal gula stugor utrustade med en eller flera balkonger har just byggts. Dessa kostar 250 kr extra för varje balkong oavsett dag. Balkonghyran läggs på efter att ev. helgtillägg har räknats ut. Definiera klassen ”MedBalkong” genom arv på lämpligt sätt.

(3 p)

Bilaga

```
// List class interface (á la Weiss)
//
// Access is via ListItr class
//
// ******************PUBLIC OPERATIONS*********************
// bool IsEmpty( )        --> Return true if empty; else return false
// bool IsFull( )         --> Return true if full; else return false
// void MakeEmpty( )      --> Remove all items
// ******************ERRORS********************************


// ListItr class interface; maintains "current position"
//
// CONSTRUCTION: with (a) List to which ListItr is permanently
//     bound or
//
// ******************PUBLIC OPERATIONS*********************
// void Insert( Etype X ) --> Insert X after current position
// bool Remove( Etype X ) --> Remove X
// bool RemoveNext()      --> Remove next cell
// bool Find( Etype X )   --> Set current position to view X
// bool IsFound( Etype X )--> Return true if X would be found
// void Zeroth( )         --> Set current position to prior to first
// void First( )          --> Set current position to first
// void operator++        --> Advance (both prefix and postfix)
// bool operator+( )      --> True if at valid position in list
// Etype operator( )      --> Return item in current position




// Queue class interface
//
// Etype: must have zero-parameter constructor and operator=
// CONSTRUCTION: with (a) no initializer;
//     copy construction of Queue objects is DISALLOWED
//
// ******************PUBLIC OPERATIONS*********************
// void Enqueue( Etype X )--> Insert X
// void Dequeue( )        --> Remove least recently inserted item
// Etype Front( )         --> Return least recently inserted item
// int IsEmpty( )         --> Return 1 if empty; else return 0
// int IsFull( )          --> Return 1 if full; else return 0
// ******************ERRORS********************************
// Predefined exception is propagated if new fails
// EXCEPTION is thrown for Front or Dequeue on empty queue
```