Uno.Holmer@chl.chalmers.se

Lösningsförslag till tentamen*

Kursnamn	**Objektorienterad programutveckling**
Tentamensdatum	**2001-01-11**
Program	**DAI 2**
Läsår	**99/00, lp III/IV**
Examinator	**Uno Holmer**

* se även `www.chl.chalmers.se/~holmer/` där finns det mesta av kursmaterialet.

Uppgift 1 (10 p)

Ingen lösning ges. Se kurslitteraturen.

Uppgift 2 (6 + 4 p)

a) (6 p)

Utskriften blir

```
Base::F Base::G Base::H
Base::F Sub1::G Base::H
Sub2::F Sub1::G Sub2::H
```

Tabellen innehåller tre basklasspekare av statisk typ *Base som pekar på var sitt objekt av klasserna Base, Sub1 respektive Sub2. Att utskriften när F anropas för det första objektet blir som ovan är väl närmast självklart. När F anropas för det andra objektet som är av typen Sub1 anropas Base::F eftersom Sub1 ärver den utan att definiera om den, Base::F anropar sen Sub1::G eftersom pekaren har dynamisk typ *Sub1 och sist anropas den ärvda Base::H. När F anropas för det tredje objektet som har typen Sub2 anropar Sub2::F den ärvda Sub1::G som anropar Sub2::H, pekaren har ju dynamisk typ *Sub2.

b) (4 p)

Svar: c. Ett C-objekt *har* ett A-objekt som det skapar och förfogar över. Relationen är 1:0..1 eftersom A-objektet är utpekat och i princip kan ha kortare livslängd än C. Relationen till B är en association 1:1, B-objektet som ett C-objekt känner till har skapats före C-objektet och kan delas av andra, t.ex. ett D-objekt. Relationerna mellan D och övriga framgår ej av koden så alla är möjliga.

Uppgift 3 (10 p)

```
template <class Etype>
class ExtendedStack : public Stack<Etype> {
public:
    void Reverse();
};

template <class Etype>
void ExtendedStack<Etype>::Reverse( ) {
    ExtendedStack<Etype> Tmp;
    while ( ! IsEmpty() ) {
        Tmp.Push( Top() );
        Pop();
    }
    *this = Tmp;
}
```

Uppgift 4 (8+8 p)

a) (8 p)

```
void Game::Run() {
    Reset();
    do {
        UserMove();
        if ( Status == Undecided )
            ComputerMove();
    } while ( Status == Undecided );

    switch ( Status ) {
        case UserWin:
            cout << "The user wins" << endl;
            break;
        case ComputerWin:
            cout << "The computer wins" << endl;
            break;
        case Draw:
            cout << "Draw" << endl;
            break;
    }
}
```

b) (8 p)

```
class Sticks : public Game {
public:
    void Reset() {
        Game::Reset();
        SticksLeft = 21;
    }

    void UserMove() {
        cout << "There are " << SticksLeft << " sticks left"
             << endl;
        cout << "You can take at most ";
        int Max = SticksLeft == 1 ? 1 : 2;
        cout << Max << " stick" << ( Max == 2 ? "s" : "" )
             << endl;
        int N;
        do {
            cin >> N;
            if ( N < 1 )
                cout << "To few, try again: ";
            if ( N > Max )
                cout << "To many, try again: ";
        } while ( N < 1 || N > Max );
        SticksLeft -= N;
        if ( SticksLeft == 0 )
            Status = ComputerWin;
    }
```

forts.

```
    void ComputerMove() {
        // Strategy: Try to force the opponent to get 3N+1 sticks
        // if not possible take only one
        int N = 1;
        if ( SticksLeft % 3 == 0 )
            N = 2;
        cout << "The computer takes ";
        cout << N << " stick" << ( N > 1 ? "s" : "" ) << endl;
        SticksLeft -= N;
        if ( SticksLeft == 0 )
            Status = UserWin;
    }
private:
    int SticksLeft;
};
```

Uppgift 5 (p)

a) (3 p) En av Grön, Gul, Röd räcker för poäng.

```
class Groen : public Stuga {
public:
    Groen( long Nr, float Gpris ) : Stuga( Nr, Gpris ) {}
    float Dagspris() { return Grundpris; }
};

class Gul : public Stuga {
public:
    Gul( long Nr, int Gpris ) : Stuga( Nr, Gpris ) {}
    float Dagspris();
};

float Gul::Dagspris() {
    if ( Dag() == Lor || Dag() == Son )
        return Grundpris*1.2;
    else
        return Grundpris;
}

class Roed : public Stuga{
public:
    Roed( long Nr, float Gpris ) : Stuga( Nr, Gpris ) {}
    float Dagspris();
};

float Roed::Dagspris() {
    if ( Dag() >= Man && Dag() <= Fre )
        return (Grundpris - 100)*0.7;
    else
        return Grundpris;
}
```

b) (5 p)

```
float Intaekt( const List<Stuga*> &Stugby ) {
    float Summa = 0;
    ListItr<Stuga*> Itr(Stugby);
    for ( Itr.First(); +Itr; Itr++  )
        Summa += Itr()->Dagspris();
    return Summa;
}
```

c) (3 p)

```
void main() {
    List<Stuga*> Stugby;
    ListItr<Stuga*> Itr(Stugby);
    Itr.Insert( new Groen(1,1495) );
    Itr.Insert( new Gul(2,1995) );
    Itr.Insert( new Roed(3,995) );
    Itr.Insert( new MedBalkong(4,1995,2) );
    cout << "Total intäkt: " << Intaekt( Stugby ) << endl;
}
```

d) (3 p)

```
class MedBalkong : public Gul {
public:
    MedBalkong( long Nr, int Gpris, int AntBalk )
        : Gul( Nr, Gpris ), AntalBalkonger(AntBalk) {}
    float Dagspris() { return Gul::Dagspris() +
                                  AntalBalkonger*250;
    }
protected:
    int AntalBalkonger;
};
```