

Datum: lördag måndag 15 oktober, 8-12 (rättning efteråt)
 Hjälpmedel: En eller två valfria läroböcker om C++
 Tid: 4 timmar

Salssida (H/V): V

Skriv inte ditt namn på tentan. Under tentan kommer ett id-nummer att delas ut och skrivas på tentan. Anteckna ditt id-nummer på separat papper. När du lämnar in tentan ska du säga vad du heter och prickas av. Tentan rättas i en första omgång två och två av er själva en timme efter tentans slut. Du måste komma tillbaka för att rätta! Skriv svaren direkt på tentan. Om du inte avser få tentan rättad kan du lämna in blankt. Grova fel ger underkänt. Gör ett försök på alla uppgifter. Tentaresultat anslas inte längre på anslagstavla i enlighet med ny policy från KTH centralt. Lycka till!

1. regler

Det har under hösten debatterats om rättskandaler och jäv. Huruvida domstolens funktion varit helt objektiv har tilldelats intresse. Konstruktiv diskussion har det varit dåligt med, snarare det motsatta. I ett rättssammhälle är det viktigt att lagar och regler följs. Det finns en del regler som är viktiga att känna till i C++ också.

- (a) I gamla standarden var det viktigt att känna till "rule of three". Tre speciella metoder som genererades åt dig även om du inte definierade dem själv. Räkna upp de tre metoderna.

- copy constructor
- Assignment operator
- Destructor.

- (b) Hur är det med default-konstruktorn när genereras den automatiskt?

A default constructor is generated every time we create an object, if user don't define the constructor, the compiler will do it automatically.

- (c) I lab1 skrevs en move-konstruktur. Hur använder man en sådan?

Our own class can benefit from being able to be moved. The move constructor has an initial parameter that is a reference to the class type which is an rvalue reference. It must ensure that the moved-from object is left in a state such that destroying that object will be harmless. Move constructor does not allocate any new memory, it takes over the memory in the given class. Move don't throw any exception but we specify "noexcept" on our constructor.

- (d) Vad kallas en klass som definierar operatorn operator()? Hur kan man använda den operatorn?

Visa med ett eget kort kodexempel.

We call a function using the call operator(). Class that defines an overloaded function call operator called: Callable Class. Classes that overload the call operator allow objects of its type to be used as if they were a function. As a simple example, the following class named absInt, has a call operator that returns the absolute value of its argument.

struct absInt {

```
int operator()(int val) const {
    return val < 0 ? -val : val;
}
```

```
int i = -42;
```

```
absInt absObj; // object
```

```
int ui = absObj(i); // passes i to absObj. operator()
```

Note: we use the call operator by applying an argument list to an absInt object in a way that looks like a function call

2. generics

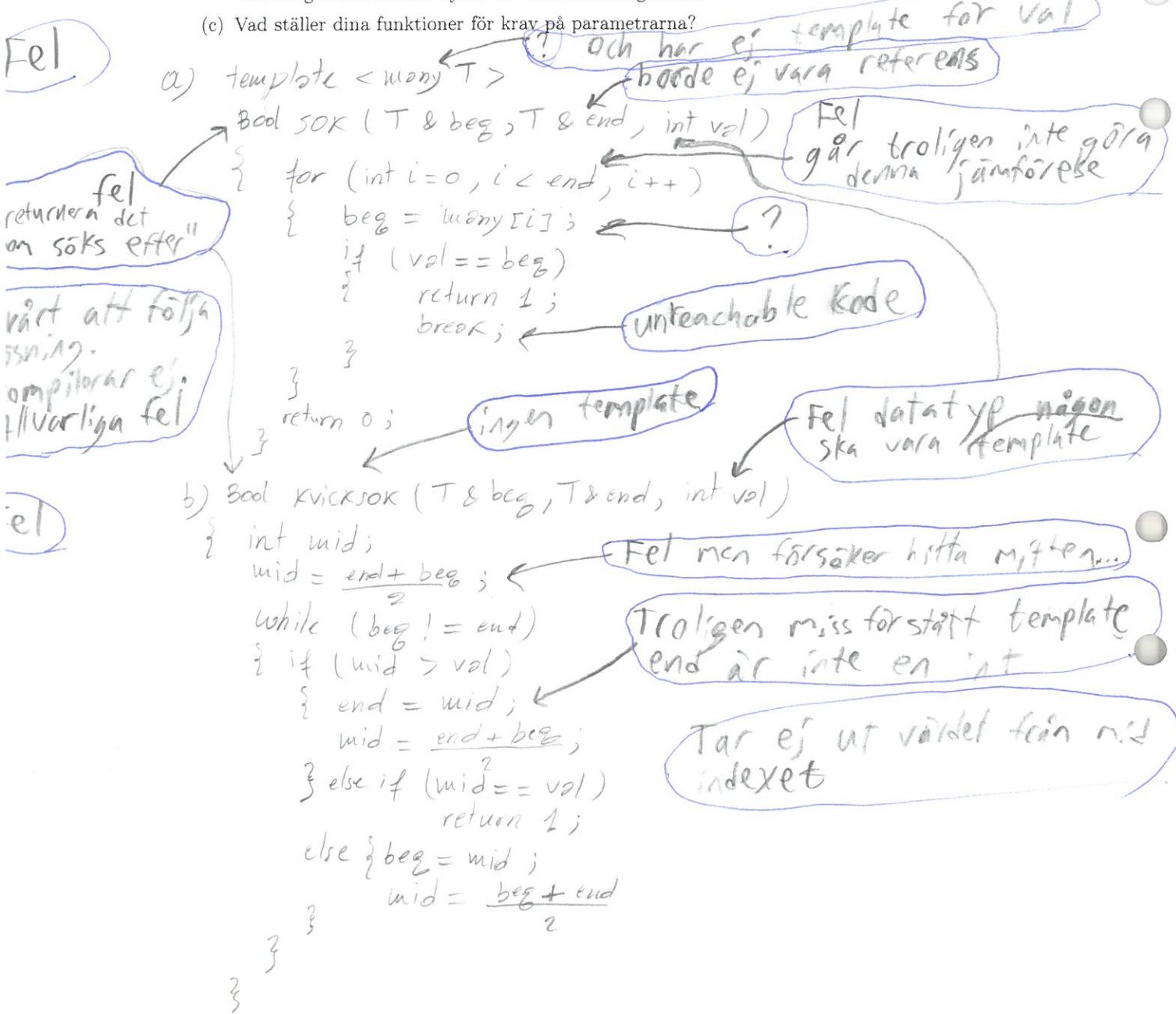
- (a) Mytomanen Tom Kvick har friats för en rad uppmärksammade dåd. Fåraherden Lamm-Bert som har ett hobbyintresse för juridik menar att Tom Kvick inte har friats, eller i alla fall inte till 100%. Lamm-Berts huvudsakliga argument är att Tom Kvicks vetskaps om huruvida en viss skogsgrind eventuellt var öppen en sommardag utgör ett beaktningsvärt indicium av skuld. Att man inte funnit något av bevisvärde i sagda skog, trots att man tömt en hel tjärn bekymrar inte Lamm-Bert det minsta. Gör följande.

Skriv en **typparametrerad** sökfunktion **sök** som tar **tre parametrar**. De första två är av en typ och definierar en mängd. Den tredje är av en annan typ. Funktionen ska söka efter den tredje parametern i den givna mängden och returnera det som söks efter, om det finns i mängden. Exakt vad som returneras får du avgöra själv. Rimliga returvärden godkännes.

- (b) Det var väldigt jobbigt att leta när allt var osorterat men man hittade faktiskt en millimeterstor bit masonit som man påstod var skelettben.

Antag att mängden är sorterad. Implementera funktionen **kvicksök** som med samma parametrar som tidigare söker med **binärsökning**. Algoritmen är ungefärlig: Bestäm mittpunkten på intervallet, om mittpunktens värde är det vi söker så är vi klara. Om värdet är större än det vi söker flytta upp lägre intervallgränsen annars flytta ner övre intervallgränsen.

- (c) Vad ställer dina funktioner för krav på parametrarna?



3. const, referenser

Det var mycket som var konstigt med Kvick-fallen. Förläggningen till varför Kvick begick brott låg i en, enligt psykologisk expertis, traumatisk barndom. Konstigt nog fick inte tingsrätten höra syskonen som motsade denna bild. Kvick hade inte körkort och kunde inte köra bil och begå brott 30 mil bort som han sagt. En gång sade han sig blivit skjutsad sisådär 170 mil under ett hektiskt brottsdygn men hans påstådda chaufför hade vattentätt alibi vilket det aldrig refererades till. Att ha koll på const och referenser är viktigt i C++!

Vad blir det för fel nedan? Var noga med const.

```
01 // ...
02 template <class T>
03 struct Polis {
04     // ...
05     vector<T> get_slask(const Polis & ref) const {
06         if (ref.check()) {
07             hide();
08         }
09         return *material;
10     }
11     bool check() {
12         if (material == nullptr) return true;
13         else return false;
14     }
15     private:
16     void hide() {
17         if (material != nullptr && material->size() > 2)
18             material->erase(material->begin());
19     }
20     vector<T> * material;
21 };
22 }
23
24 int main() {
25     Polis<int> p;
26     // ...
27     p.get_slask(p);
28     //...
29 }
```

we cannot change the pointer "material" because ref refer to constant type!

Allvarligt

Fel!

anledningen är för att metoden ej är konstant

Svar: const-metod för
inte en typ för icke-konst
metod. check() och hide()

For safety's sake, const is used
to ensure that get_slask() cannot
change the object "ref"

man får inte skicka en const-deklarerad referensvariabel
som en rörsparametrar till en icke-const deklarerad
parametrar, men motsatsen är möjlig - Det är funktionen
som kan förenstra en variabel.

N.B.: If you have a const function, `get_slask()`, all that means is that it guarantees it won't change the object `p`. so just because it is const doesn't mean that non-const objects, like `p`, can't use it.

Rules:

- const-functions can always be called
- non-const functions can only be called by non-const objects.

4. dynamisk bindning, polymorfism, objekt

Kvick-utredningarna har kritiserats för att det ställdes ledande frågor till Kvick. Men en vittnesprofessor bedyrade att Kvick behövde hjälp att komma ihåg. Dessutom var det naturligt att han sade fel om olika objekt (t.ex. mordvapen 29 ggr) under hågkomstprocessen eftersom den "sanna" minnesbilden var för jobbig att hantera. För att få klarhet vad som händer i virtuell C++ hjälper dock en klar minnesbild!

- (a) vad skriver koden nedan ut?
- (b) Vad skrivs ut om man tar bort alla `virtual`?
- (c) Rita en minnesbild på objekten i `main`.
- (d) Beskriv en situation när det kan vara bra att använda `virtual`.

```
Base class Vittne
struct Vittne {
    Vittne() : m_influens(this) {}
    virtual void utsaga() { cout << "icke skyldig" << endl; }

    Vittne * m_influens;
};

Derived class
struct PseudovetenskapligExpert : public Vittne {
    virtual void utsaga() { cout << "skyldig" << endl; }
};

void granskning(Vittne x) { // OBS x by value
    x.utsaga();
}

void debattinlagg(Vittne & x) { // OBS x by reference
    x.utsaga();
}

int main()
```

Dynamisk Bindning

```

    PseudovetenskapligExpert sven;
    Vittne gun = sven;           // OBS copy

    sven.utsaga();   // skyldig
    gun.utsaga();   // icke skyldig

    sven.m_influens->utsaga(); // skyldig
    gun.m_influens->utsaga(); // skyldig

    granskning(sven); // icke skyldig
    debattinlagg(sven); // skyldig
}

```

a) skyldig
icke skyldig

skyldig
skyldig

icke skyldig
skyldig

skyldig
skyldig

b) icke skyldig
" skyldig
" skyldig
" skyldig
" skyldig
" skyldig
" skyldig

d) when we have different derived classes that answer in different ways to the same call function,

9 Fel

Rätt

Rätt

Rätt



Fyll i ferkoder i tabellen nedan för varje fel på varje uppgift. Ferkoden är fyra siffror Xx-YY där X är frågenummer, x delfråga och YY ett löpnummer (01, 02 ...). Markera felet på tentan i marginalen (samma fel kan härledas till två ställen ibland) och skriv utförlig kommentar i tabellen nedan samt, om lämpligt, kort kommentar på tentan (använd annan färg t.ex. blå eller rött bläck som finns att låna). Underlättelse att rätta små som allvarliga fel kan påverka resultatet på rättarens egen tenta. Om du är osäker på om felet är allvarligt, ringa in ferkoden. Om det saknas plats, använd kringliggande rutor som är utan anmärkningar. Man kan biktta egna fel som man kommer på.

| Fråga | allvarlig ferkod | övriga fel | kommentar |
|-------|------------------|------------|---|
| 1 | | | c) har ej visat hur den används bara hur den implementeras. |
| 2 | | | a) Returnerar ej elementet Buggar, där ej två template typer kanstig sak med many b) massor med buggar använder referenser som argument etc. Se tentan |
| 3 | | | Har inte förstått konsekvensen av kanst med funktioner. |
| 4 | | | b) Fel c) tomt |
| Bikt | | | |

Rättat av id 092 H

Rättat av id 159 H

009

✓



Family name, first name

Personal Registration Number

Programme

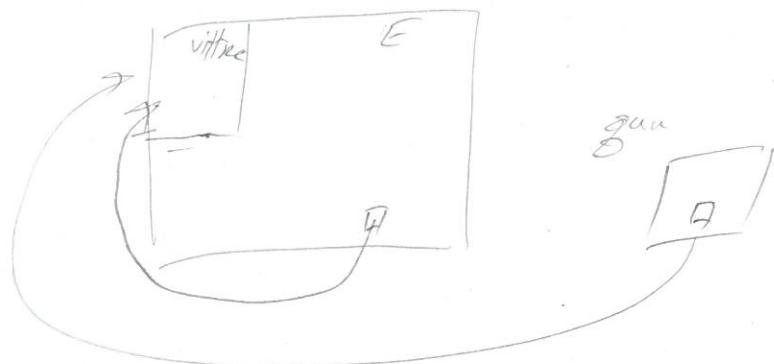
Sheet no.

Problem no.

- callable object p571.
- functor ?
- binary search

- skyldig
icke
sky
skyldig
icke
skyldig

skyldig
icke
=
||
||
||
||







Family name, first name

Personal Registration Number

Programme

Sheet no.

Problem no.

SÖK (T_a, T_b, P_c) T definieras en
mångd.a) template < meny T >
utanet

```
int SÖK (&T const int
          &beg, &end, val)
{ for (int i=0, i< end, i++)
    if TELL beg = T[i]; val == beg
        return True 1
    else
        return -1
    return 0;
```

SÖK söks efter
den c ~~sek~~; den
givna mångden T
och return c, om
det finns i T

Bool ~~int~~ knick ($T \& beg, T \& end, int val$)

```
{ T mid
    mid = end + beg / 2;
    while (end != beg) {
        if (mid > val)
            mid = beg +
            end = mid;
            mid = beg + end / 2;
        else if mid == val
            return 1;
        else beg = mid
            mid = beg + end / 2;
```



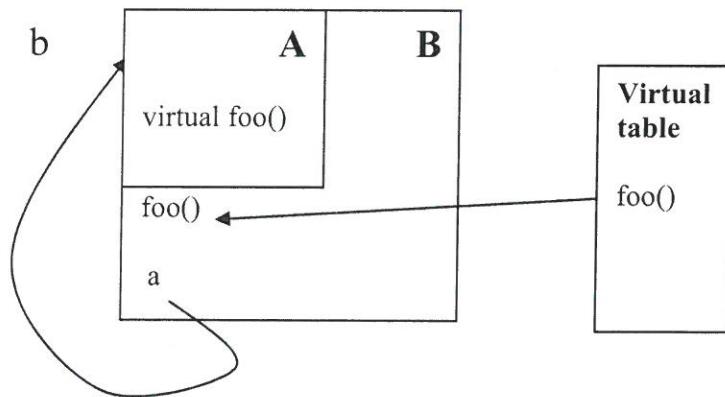
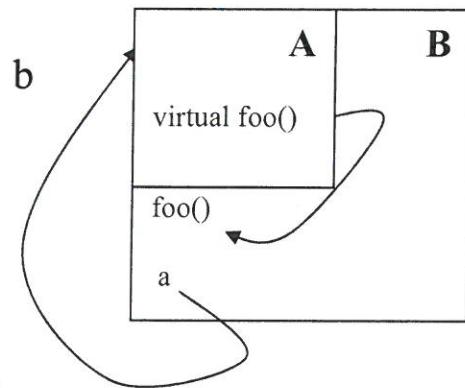
Rita minnesbild på instanser

```
struct A {  
    virtual void foo() { std::cout << "J"; }  
};  
  
struct B : public A {  
    B() : a(this) {}  
    void foo() { std::cout << "A"; }  
    A *a;  
};
```

Om jag instansierar ett b

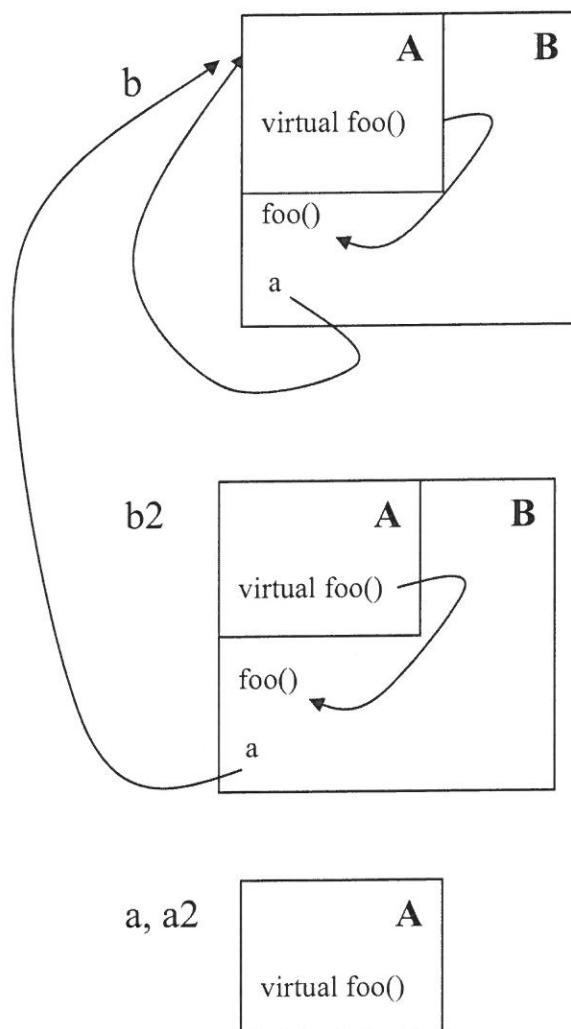
```
B b;
```

Nedan två förslag på ritning. Man kan även rita klasserna ovanpå varandra istället för inuti som jag brukar göra.



```
B b;  
B b2 = b;  
A a;  
A & a2 = a;
```

Man måste rita en instans för varje objekt som instansieras!



Datum: Tisdag 18 oktober 2011, 8-12 (rättning ca 12:40-14)

Salssida (H/V): _____

id: _____

Hjälpmedel: En eller två valfria läroböcker om C++

Tid: 4 timmar

Skriv inte ditt namn på tentan. Under tentan kommer ett id-nummer att delas ut och skrivas på tentan. Anteckna ditt id-nummer på separat papper. När du lämnar in tentan ska du säga vad du heter och prickas av. Tentan rättas i en första omgång två och två av er själva en timme efter tentans slut. Du måste komma tillbaka för att rätta! Skriv svaren direkt på tentan. Om du inte avser få tentan rättad kan du lämna in blankt. Grova fel ger underkänd. Gör ett försök på alla uppgifter. Tentaresultat anslås inte längre på anslagstavla i enlighet med ny policy från KTH centralt. Lycka till!

1. Konstiga hyresbidrag

Några riksdagsledamoter har svårt att förstå riksdagens regler för hyresbidrag. Ja inte bara politiker utan eliten av krönikörer och ledarskribenter verkar ha svårt att förstå vad som gäller. Särskilt svårt blir det när informationen om sagda regler plötsligt ändras vilket inte ger intryck av varaktiga och konstanta regler. Det är mycket viktigt att förstå vad som är konstant och varaktigt och vad som inte är det i C++.

Koden nedan kommer inte att kompileras. Programmeraren har inte förstått `const`. Gå igenom koden och förklara de två kompileringsfelen. Ange berörda rader!

```
01 struct A {  
02     void hyra(A & a) const {  
03         };  
04 };  
05  
06 struct B : public A {  
07     void blankett(const B & b, A a) {  
08         a.hyra(b);           // what is the problem?  
09     }  
10 };  
11  
12 struct C : public B {  
13     A blankett(const A & a) {  
14         return a;  
15     }  
16 };  
17  
18  
19 void fyll_i_blankett(const A & a, B b, const C & c) {  
20     b.blankett(b, c.blankett(a)); // what exactly is the problem?  
21 }
```

2. Dynamiska händelser

Vad är det som hänt, händer eller borde hända? Frågan är om vi någonsin får någon klarhet i riksdagens arbetsmetoder. För att programmera C++ är det viktigt att kunna förstå vad som händer i metodanrop. Speciellt om några är virtuella.

- (a) Vad skriver följande kod ut?
- (b) Vad skulle skrivas ut om alla virtual togs bort?
- (c) Vilka av de fem print-anropen använder inte dynamisk bindning för att i runtime binda funktionsanropen? Motivera!

```
#include <iostream>
using namespace std;
struct A {
    A(string s) : name(s) {};
    string name;
    virtual void print() { cout << name << " betalar hyran" << endl; };
};
struct B : A {
    B(string s) : A("Riksdagen"), name(s) {};
    string name;
    virtual void print() { cout << name << " betalar hyran " << endl; };
};

void f1(A & p) {
    p.print();
}
void f2(A p) {
    p.print();
}
int main() {
    B b1("Håkan");
    A & aref(b1);
    A acopy = b1;

    b1.print();
    aref.print()
    acopy.print();
    f1(b1);
    f2(b1);

}
```

Svar c) _____

3. Fibonaccidrevet

Många politiker, ja inte bara politiker, är upprörda över mediedrev, i alla fall när man själv är utsatt för ett. En del verkar definiera mediedrev som upprepadet av lögner. Andra menar att lögnerna spär på varandra ungefär som en fibbonacciserie, att nästa lön är summan av de två föregående. I vilket fall verkar mångas åsikt vara att mediedreven inte fyller någon vettig funktion i samhället objektivt sett.

- (a) Nedanstående kod använder sig av en snillrik operator. Denna `operator()` eller snarare klasser/objekt som använder en sådan operator har ett speciellt namn, vilket?

```
// generate fibonacci sequence beginning with 0 and 1 and each subsequent
// number is the sum of the previous two: 0 1 1 2 3 5 8 11 19 ...
struct generate_fibonacci {
    generate_fibonacci() : m_nFirst( 0 ), m_nSecond( 1 ) {
    }

    int operator() () {                                // snillrik operator
        int nNew = m_nFirst + m_nSecond;
        m_nFirst = m_nSecond;
        m_nSecond = nNew;
        return nNew;
    }

private:
    int m_nFirst;
    int m_nSecond;
};
```

- (b) Sätt dig in i koden. Skriv ett kort main program som skriver ut de sju första fibonacci-talen.

```
#include <iostream>
using namespace std;
int main()
```

4. Quicksort

I dagens mediasamhälle är det viktigt att snabbt kunna sälla och sortera bland all information som flyger omkring. Quicksort använder sig av partitionering som givet ett pivot-element sorterar alla viktigare element till höger och oviktigare till vänster om pivotelementet.

- (a) Skriv en typparametrerad partition-kod så att godtycklig klass som implementerat `operator<` sorteras. Utgå från nedanstående variant som väljer första elementet i vektorn som pivotelement.

```
void swap(int & a, int & b) {
    int tmp = a;
    a = b;
    b = tmp;
}

int * partition(int vek [], int nrOfElements)
{
    int * tmp_pivotidx = vek + nrOfElements - 1;
    swap(*vek, *tmp_pivotidx); // Move pivotelement to end

    int * left=vek;           // Move all less than pivot to the left
    for(int * it=vek; it!=vek + nrOfElements - 1; ++it) {
        if(*it <= *tmp_pivotidx) {
            swap(*left, *it);
            ++left;
        }
    }
    swap(*tmp_pivotidx, *left); // swap back pivot element

    return left;             // return the position where the pivot value is stored
                            // which is in its correct (sorted) position
}
```

Man ska kunna anropa partition med följande kod, observera parametertyperna.

```
int main() {
    //...
    partition(b.begin(), v.end());           // Två iteratorer
    partition(array, array + nrElements);     // Två pekare
}
```

Körexempel:

```
13 28 10 16 7 4 12 1 19 23 11 8 3 ->
3 10 7 4 12 1 11 8 13 23 28 16 19
```

Paris London Stockholm Berlin Oslo Rome Madrid Tallinn Amsterdam Dublin ->
Dublin London Berlin Oslo Madrid Amsterdam Paris Tallinn Rome Stockholm

Ett körexempel med utskrift (saknas i koden) efter varje swap

```
13 28 10 16 7 4 12 1 19
19 28 10 16 7 4 12 1 13      Spara undan pivotvärdet 13
10 28 19 16 7 4 12 1 13
10 7 19 16 28 4 12 1 13
10 7 4 16 28 19 12 1 13
10 7 4 12 28 19 16 1 13
10 7 4 12 1 19 16 28 13
10 7 4 12 1 13 16 28 19      Lägg tillbaka pivotvärdet
```

- (b) Förutom `operator<` vad ställer din kod för krav på parametrarna?

svar 4a)





Datum: lördag 11 februari 2012, 14-18 (rättning efteråt)

Salssida (H/V): _____

id: _____

Hjälpmedel: En eller två valfria läroböcker om C++

Tid: 4 timmar

Skriv inte ditt namn på tentan. Under tentan kommer ett id-nummer att delas ut och skrivas på tentan. Anteckna ditt id-nummer på separat papper. När du lämnar in tentan ska du säga vad du heter och prickas av. Tentan rättas i en första omgång två och två av er själva en timme efter tentans slut. Du måste komma tillbaka för att rätta! Skriv svaren direkt på tentan. Om du inte avser få tentan rättad kan du lämna in blankt. Grova fel ger underkänt. Gör ett försök på alla uppgifter. Tentaresultat anslås inte längre på anslagstavla i enlighet med ny policy från KTH centralt. Lycka till!

1. rule of three

En matlabkopieprodukt har implementerat matriser som kan hantera godtyckligt stora matriser. Vilka tre metoder är det viktigt att implementera i den matris-klassen?

Kopiekonstrutor, Destruktor, tilldelningsoperatör

2. generics

Skriv en typparametriserad funktion som tar två likadana typparametriserade parametrar. Parametrarna definierar en mängd. Funktionen ska kunna anropas både med två pekare till en array eller med två iteratorer till en typparametriserad container t.ex. vector/map m.m.

Funktionen ska returnera **true** om två på varandra följande lika element påträffas, annars returnera **false**.

Algoritmen skulle kunna vara ungefär; sätt en variabel previous till startparametern och stega den senare ett steg. Iterera över mängden och jämför elementen de hänvisar till samt stega dem bågge i varje varv. Returnera sannt så fort två på varandra följande lika element påträffas. Det är inte ett allvarligt fel att göra fel vid gränserna dåremot ska det uppmärksammas vid kamraträttning.

3. call by value, reference and const

I C++ kan man anropa en funktion **by value** eller **by reference**. Man kan dessutom **const**-deklarera parametrarna.

- (a) Får man skicka en **const** deklarerad referensvariabel som anropsparameter till en icke-**const** deklarerad referensparameter? (rad 23) *Nej*
- (b) Får man skicka en icke-**const** deklarerad referensvariabel som anropsparameter till en **const** deklarerad parameter? (rad 28) *J*
- (c) Spelar det i ovanstående frågor någon roll om parametern är deklarerad som **call by value** (rad 17 och rad 18)? *J*
- (d) Rad 15 och rad 25 anropar sin funktion med siffran 17 som parameter. Den kommenterade raden 20 kompilerar däremot inte. Varför?

```
01 struct F {  
02  
03     void a(int k) {}  
04     void b(int & k) {}  
05     void c(const int & k) {}  
06  
07 };  
08 int main() {  
09     int x = 5;  
10     int & xref = x;  
11     const int & const_x_ref = x;  
12  
13     F f;  
14  
15     f.a(17);  
16     f.a(x);  
17     f.a(xref);  
18     f.a(const_x_ref);  
19  
20     // f.b(17);  
21     f.b(x);  
22     f.b(xref);  
23     f.b(const_x_ref);  
24  
25     f.c(17);  
27     f.c(x);  
28     f.c(xref);  
29     f.c(const_x_ref);
```

- (e) Man kan **const**-deklarera en metod också vilket gör att kompilatorn kan protestera mot vissa anrop. Varför kan man vilja **const**-deklarera metoder?
- (f) Definiera en tom **const**-deklarerad metod i klassen F ovan. Skriv minimal kod som genererar kompilexceptionsfel p.g.a. att metoden är **const**-deklarerad samt skriv svaret på förra frågan här under.

4. dynamisk bindning, polymorfism

Dynamisk bindning kallas det när man skriver anropskod som inte kan avgöras i compile-time. Det är först i runtime som det avgörs vilken kod som kommer att köras. Skriv minimal kod som illustrerar ett anrop med dynamisk bindning.

I vissa böcker kan det skrivas om detta under polymorfism och/eller så står det under ett annat keyword som jag inte vill berätta utan jag vill att ni ska komma på själva.

5. minneshantering

Följande ofullständiga kod är en delmängd av koden från ett äventyrsspel. Alla kod är inte återgiven.

Koden har problem med hanteringen av dynamiskt minne. Beskriv minneshanteringsproblemen i metoderna *killPeasant* och *load*. Metoderna *clear* och *erase* finns i `std::vector` och bör stå beskrivna i er bok.

```
#include <vector>
#include <string>
using namespace std;

class ClassWithAllTheCode {
    vector<Room *> rooms;
    vector<GameObject *> people; // GameObject är basklass
    vector<GameObject *> items;
public:
    ClassWithAllTheCode() {
        rooms.push_back(new Room(1, 2, 3)); // initierar alla rumsförbindelser
        rooms.push_back(new Room(1));
        //...
        people.push_back(new Peasant(rooms[1])); // sätt ut folk och föremål
        people.push_back(new Soldier(rooms[3]));
        //...
        items.push_back(new Sword(rooms[3]));
        //...
    }
    void killPeasant(GameObject * g) {
        // Ta bort bonden från people-vektorn
        vector<GameObject *>::iterator i;
        for (i = people.begin(); i < people.end(); i++) {
            if (g == *i) {
                people.erase(i);
            }
        }
    }
    void load(GameFile file) {
        GameObject * p;
        people.clear();
        while (p = file.getPeople()) {
            people.push_back(p);
        }
        items.clear();
        while (p = file.getItem()) {
            items.push_back(p);
        }
    }
    // fler metoder med logik, AI, interagerande med spelaren m.m.
    // ...
    void run() { /* ... */ };
}
```

Båda *erase*, och *clear*,
tar bort påkore ur
vektorn men *delete*
görs inte.

Fyll i felkoder i tabellen nedan för varje fel på varje uppgift. Felkoden är fyra siffror Xx-YY där X är frågenummer, x delfråga och YY ett lopnummer (01, 02 ...). Markera felet på tentan i marginalen (samma fel kan härledas till två ställen ibland) och skriv utförlig kommentar i tabellen nedan samt, om lämpligt, kort kommentar på tentan (använd annan färg t.ex. blå eller rött bläck som finns att låna). Underlättelse att rätta små som allvarliga fel kan påverka resultatet på rättarens egen tenta. Om du är osäker på om felet är allvarligt, ringa in felkoden. Om det saknas plats, använd kringliggande rutor som är utan anmärkningar.

| Fråga | allvarlig felkod | övriga fel | kommentar |
|-------|------------------|------------|-----------|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

Rättat av id _____

Rättat av id _____

1. Att söka sekvenser

- (a) FRA-lagen är på tapeten igen. Man ska kunna söka igenom delar av internet efter delmängder av ännu inte helt definierade sökmängder. Detaljerna i denna "security search" är höjt i dunkel.

Skriv en generell typparametriserad funktion SeqSearch som tar fyra parametrar. Parametrarna definerar två sekvenser i godtycklig mängd. Funktionen ska returnera huruvida den andra sekvensen är en delsekvens av den första. Exempel: Exempel: $\{1\ 2\ 3\}$ är en delsekvens av $\{1\ 2\ 3\ 4\}$ och $\{2\ 4\}$ är en delsekvens av $\{1\ 2\ 3\ 4\}$ medan varken $\{3\ 2\}$ eller $\{2\ 2\}$ är en delsekvens av $\{1\ 2\ 3\ 4\}$. Sökning efter tomma mängden ska ge falskt svar.

Algoritmen är ungefär, gå igenom det stora intervallet och kontrollera om ett element även finns i delsekvensen. Sökningen är sann om alla delsekvensens element påträffats i ordning.

- (b) Vad ställer din lösning för krav på parametrarna?
(c) Kan man skriva funktionen så att den anropas med två olika sorters typparametrar? Motivera ditt svar.

- polymorfi, ärvt
- run-time polymorf, compile-time polymorfism
- multipelt arv.
- Input / output i C
- Input / output i C++
- Effektiv C++ (David Abrahams)



```

#include <string>
#include <iostream>
using namespace std;

// correct solution
template<class It1, class It2>
bool subseq(It1 it1, It1 end1, It2 it2, It2 end2)
{
    if (it2 == end2) return false;
    for(; it1 != end1; ++it1)
        if(it2 != end2 && *it1 == *it2)
            it2++;
    return it1 == end1 && it2 == end2;
}

// incorrect solution attempts
template<class It1, class It2>
bool subseq1(It1 it1, It1 end1, It2 it2, It2 end2)
{
    if (it2 == end2) return false;
    for(; it1 != end1; ++it1) {
        if(it2 != end2 && *it1 == *it2) {
            while (it1 != end1 && it2 != end2 && *it1 == *it2) {
                ++it1;
                ++it2;
            }
        }
    }
    return it1 == end1 && it2 == end2;
}

template<class It1, class It2>
bool subseq2(It1 it1, It1 end1, It2 it2, It2 end2)
{
    if (it2 == end2) return false;
    for(; it1 != end1; ++it1) {
        if(it2 != end2 && *it1 == *it2) {
            It2 i = it2;
            for (; i != end2; ++i, ++it1) {
                if (*i != *it1) break;
            }
            if (i == end2) return true;
        }
    }
    return false;
}

template<class It1, class It2>
bool subseq3(It1 * it1, It1 * end1, It2 * it2, It2 * end2)
{
    if (it2 == end2) return false;
    for(; it1 != end1; ++it1) {
        if(it2 != end2 && *it1 == *it2) {
            It2 * i = it2;
            for (; i != end2; ++i, ++it1) {
                if (*i != *it1) break;
            }
            if (i == end2) return true;
        }
    }
    return false;
}

template<class It1, class It2>
bool subseq4(It1 * it1, It1 * end1, It2 * it2, It2 * end2)
{
    if (it2 == end2) return false;
    for(; it1 != end1; ++it1) {
        if(it2 != end2 && *it1 == *it2) {
            It2 * i = it2;
            for (; i != end2; ++i, ++it1) {
                if (*i != *it1) break;
            }
            if (i == end2) return true;
        }
    }
    return false;
}

```

(1)

(2)

(3)

(4)



```

template<class It>
bool subseq5(It * it1, It * end1, It * it2, It * end2)
{
    for(; it1 != end1; ++it1)
    {
        if(it2 != end2 && *it1 == *it2)
            it2++;
    }
    return it1 == end1 && it2 == end2;
}

template<class T>
bool subseq6(T s1, T s2) { // Lösung zu P200K
    unsigned int x = 0;
    for (unsigned int i = 0; i < s1.size(); i++) {
        if (s1[i] == s2[i]) {
            x++;
        }
    }
    return x == s2.size();
}

template<class It>
bool subseq7(It it1, int end1, It it2, int end2)
{
    for(int i = 0; i < end1; ++i, ++it1)
    {
        if (*it1 == *it2) {
            for(int j = 0; j < end2 && (*it1 + j) == *(it2 + j); ++j)
                if (j == end2) return true;
        }
    }
    return false;
}

template<class It>
bool subseq8(It * it1, int end1, It * it2, int end2)
{
    for(int i = 0; i < end1; ++i, ++it1)
    {
        if (*it1 == *it2) {
            int j = 0;
            for(j = 0; j < end2 && (*(it1 + j) == *(it2 + j)); ++j)
                ;
            if (j == end2) return true;
        }
    }
    return false;
}

void teststring(string s) {
    string ipr = "ipred";
    cout << "subseq: '" << ipr << "' << s << "' " <<
    subseq(s.begin(), s.end(), ipr.begin(), ipr.end()) << endl;
    cout << "subseq1: '" << ipr << "' << s << "' " <<
    subseq1(s.begin(), s.end(), ipr.begin(), ipr.end()) << endl;
    cout << "subseq2: '" << ipr << "' << s << "' " <<
    subseq2(s.begin(), s.end(), ipr.begin(), ipr.end()) << endl;
    cout << "subseq3: '" << ipr << "' << s << "' " <<
    subseq3(s.c_str(), s.c_str() + 15, ipr.c_str(), ipr.c_str() + 5) << endl;
    cout << "subseq5: '" << ipr << "' << s << "' " <<
    subseq5(s.c_str(), s.c_str() + 15, ipr.c_str(), ipr.c_str() + 5) << endl;
    cout << "subseq6: '" << ipr << "' << s << "' " <<
    subseq6(s, ipr) << endl;
    cout << "subseq7: '" << ipr << "' << s << "' " <<
    subseq7(s.begin(), s.size(), ipr.begin(), ipr.size()) << endl;
    cout << "subseq8: '" << ipr << "' << s << "' " <<
    subseq8(s.c_str(), s.size(), ipr.c_str(), ipr.size()) << endl;
    // cout << "subseq4: '" << s << subseq4(s.c_str(), s.c_str() + 12, ipr.c_str(), ipr.c_str() + 5) <<
    endl; // does not compile
}

int main() {
    string s1 = "hejhopp ";
    string s2 = "ipred ";
    string s3 = "iföldipred ";
    teststring(s1);
    teststring(s2);
    teststring(s3);
}

```



```
/*
subseq: 'ipred' i 'hejhopp'      ' 0
subseq1: 'ipred' i 'hejhopp'      ' 0
subseq2: 'ipred' i 'hejhopp'      ' 0
subseq3: 'ipred' i 'hejhopp'      ' 0
subseq5: 'ipred' i 'hejhopp'      ' 0
subseq6: 'ipred' i 'hejhopp'      ' 0
subseq7: 'ipred' i 'hejhopp'      ' 0
subseq8: 'ipred' i 'hejhopp'      ' 0
subseq: 'ipred' i ' i p r e d'   ' 1
subseq1: 'ipred' i ' i p r e d'   ' 1
subseq2: 'ipred' i ' i p r e d'   ' 0
subseq3: 'ipred' i ' i p r e d'   ' 0
subseq5: 'ipred' i ' i p r e d'   ' 1
subseq6: 'ipred' i ' i p r e d'   ' 0
subseq7: 'ipred' i ' i p r e d'   ' 0
subseq8: 'ipred' i ' i p r e d'   ' 0
subseq: 'ipred' i 'i f ö l j d ipred' ' 1
subseq1: 'ipred' i 'i f ö l j d ipred' ' 1
subseq2: 'ipred' i 'i f ö l j d ipred' ' 1
subseq3: 'ipred' i 'i f ö l j d ipred' ' 1
subseq5: 'ipred' i 'i f ö l j d ipred' ' 1
subseq6: 'ipred' i 'i f ö l j d ipred' ' 0
subseq7: 'ipred' i 'i f ö l j d ipred' ' 0
subseq8: 'ipred' i 'i f ö l j d ipred' ' 1
*/

```



Fyll i felkoder i tabellen nedan för varje fel på varje uppgift. Felkoden är fyra siffror Xx-YY där X är frågenummer, x delfråga och YY ett löpnummer (01, 02 ...). Markera felet på tentan i marginalen (samma fel kan härledas till två ställen ibland) och skriv utförlig kommentar i tabellen nedan samt, om lämpligt, kort kommentar på tentan (använd annan färg t.ex. blå eller rött bläck som finns att låna). Underlättelse att rätta små som allvarliga fel kan påverka resultatet på rättarens egen tenta. Om du är osäker på om felet är allvarligt, ringa in felkoden. Om det saknas plats, använd kringliggande rutor som är utan anmärkningar.

| Fråga | allvarlig felkod | övriga fel | kommentar |
|--------|------------------|------------|--------------------------------------|
| 1 ② | 2 - 01 | | tår endas samturhängande sibse 95 |
| 2 ③ | 2 - 01 | | |
| 3 6 | 6 - 02 | | tommars utlösden |
| 4 | | | |
| 5 | | | |

Rättat av id _____

Rättat av id _____

