UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Exam in:	**INF3110/4110 Programming Languages**
Day of exam:	**4. desember 2006**
Exam hours:	**14:30 – 17:30**

This examination paper consists of 9 pages including page 8 that is used in the answering of question 1a, and page 9 that may be used for sketching this answer.

Appendices:	**No**
Permitted materials:	**All**

Make sure that your copy of this examination paper is complete before answering.

This exam consists of 3 questions that may be answered independently. If you think the text of the questions is unclear, make your own assumptions/interpretations, but be sure to write these down as part of the answer.

Good luck!

Contents

Question 1 Runtime-systems, scoping, types (weight 40%) 2
Question 2 ML (weight 40%) 4
Question 3 Prolog (weight 20%) 6

Question 1 Runtime-systems, scoping, types (weight 40%)

1a

NB.: This part of question 1 is answered by filling in values and links in the figure at page 8, and deliver it together with the rest of the answers! Remember to fill in candidate number and date!

In the following Java-like program we use redefinition (overriding) of a virtual method `change`. Note that the class `A` is defined locally to `Circle`, and that the subclass `B` is defined locally to the method `fromAtoB`. The call of the method `super.change()` implies call of the method as it is defined in the superclass, not as redefined in `B`.

The program execution is started by execution of the `main` method in the class Program.

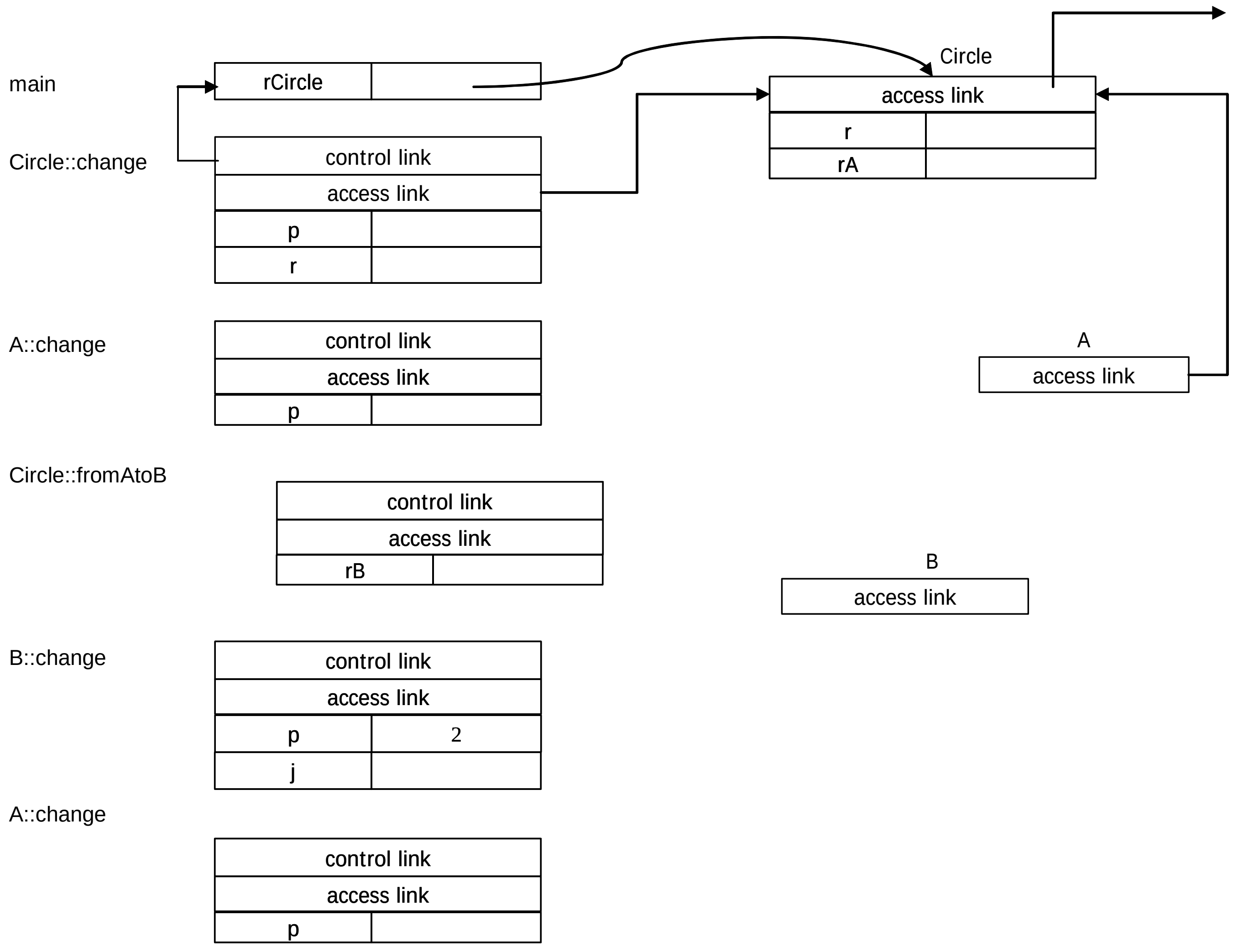
```
class Program {
  public static void main(String[] args) {
    Circle rCircle= new Circle(); rCircle.change(2);
  }
}

class Circle {
  int r;
  Circle() {r = 1;}
  class A {
    void change(int p){r = r + p;}
  }
  A rA = new A();
  void change(int p){
    int r = p;
    rA.change(p);
    fromAtoB();
    rA.change(p);     //        (*)
  }
  void fromAtoB(){
    class B extends A {
      void change(int p){
        int j = r;
        super.change(p);
        r = r + j;
      }
    }
    B rB = new B();
    rA = rB;
  }
}
```

We assume that the language has static scoping. Put in values on parameters, references, and variables, access links and control links in the activation blocks and in objects as they are when the activation record for the call `rA.change(p)` in the line marked with (*) is on top of the stack. Put in values and links as they are just before the `change` method exits. The figure below contains activation blocks that may be on the stack (which means that they will not all be there). Some of the links and references are put in already. Assume that there has been no garbage collection.

The activation block for main has for simplicity reasons no access link.

The access link for an object denotes the instance that represents the textually enclosing block (that is object for an enclosing class or activation block for an enclosing method).



1b

Assume that we have the following classes:

```
class Point {
  int x,y;
  Point(int px, int py) {x = px; y = py;}
}
class ColoredPoint extends Point{
  Color c;
  ColoredPoint(int px, int py, Color pc) {x = px; y = py; c = pc;}
}

class Circle {
  Point center;
  int radius;
  Circle(Point p, int r) {center = p; radius = r;}
  Point getCenter() {return center}
  void setCenter(Point p) {center = p;}
}
class ColoredCircle extends Circle{
  ColoredCircle(ColoredPoint cp, int r) {center = cp; radius = r;}
  ColoredPoint getCenter() {return (ColoredPoint)center}
  void setCenter(? p) {center = p;}
}
```

and two references:

```
Point somePoint;
Circle someCircle;
```

Note that the type of the parameter `p` in the last method `setCenter` shall be '`?`' – this is used in b).

a)

This language allows that a redefinition (overriding) of a method changes the return type. This has been done for the method `getCenter`. Will it be so that the statement

```
somePoint = someCircle.getCenter()
```

in all cases will be type correct and can be checked at compile time, that is for all possible values of `someCircle` (apart from the case where `someCircle` does not denote any object), or is a runtime type check required?

Give a short (3-4 sentences) explanation of the answer.

b)

Which type shall be at the point marked with '`?`' in order that the following statement

```
someCircle.setCenter(somePoint)
```

in all cases will be type correct and can be checked at compile time, that is for all possible values of `someCircle` (apart from the case where `someCircle` does not denote any object)?

Give a short (3-4 sentences) explanation of the answer.

Question 2 ML (weight 40%)

Note that your ML-function definitions must be purely functional, imperative constructs are not allowed, specifically you may not use reference types or while loops. In the definition of functions, you may use functions you have been asked to define earlier in the exercise even if you did not manage to define them.

2a Basic functions

a) Define the ML-function

```
fun max(i:int,j:int):int = ...
```

which returns the largest of i and j.

b) Define the ML-function

```
fun maxlist(il:int list):int = ...
```

which returns the largest integer in the list il. The function should return the value 0 if the list is empty.

c) Define the ML-function

```
fun sumlist(il:int list):int = ...
```

which returns the sum of the integers in the list.

2b Higher-order programming

Recall the function *map*:

```
map: (’a -> ’b) -> ’a list -> ’b
```

which has the following definition in ML:

```
fun map f[] =[]
  | map f(x::xs) =(f x) :: map f xs ;
```

map applies a function to every element of a list and returns a list of the function’s results.

Define the ML-function

```
mapFirstTwo: (’a * ’a -> ’b) -> ’a list -> ’b list
```

which works like map except that the function argument is always a function of two arguments (like the operator +). Use the first and second elements of the list argument as the first pair of arguments to the function argument, then the second and third elements, then the third and the fourth, and so on. If there are fewer than two elements in the list, the empty list should be returned. Here is an example:

```
mapFirstTwo (op +) [2,3,4,5,7] ---> [5,7,9,12]
```

2c Terms

A *term* consists of function symbols (*f*,*g*,*h*), variables (*x*,*y*,*z*) and constants (*a*,*b*,*c*). Examples of terms are: *a*(), *f* (*a*), *f* (*x*, *y*), and *f* (*g*(*a*, *b*),*h*(*c*, *d*)). Note that for the sake of simplicity we sometimes omit () from constants,i.e. we write *a* instead of *a*().

In ML, terms can be represented with the following recursive datatype:

```
datatype Term = Term of (string * (Term list));
```

a) Give ML values for the terms:

```
1.  f(a):              val fa = ...
2.  f(x,y,z):          val fxyz = ...
3.  f(g(a,b),h(c,d)):  val fgh = ...
```

The datastructure for Term is a multibranching tree. I.e. every node may have any number of children. We want to calulate the height of a tree: A tree with no children has height 1 and generally the height of a tree is 1 + the height of the highest of its children.

b) Define the ML-function

```
val height=fn: Term -> int
```

which calculates the height of a Term.

Example: For the three terms you gave values for in a), height should return 2, 2, and 3.

Question 3 Prolog (weight 20%)

The University of Oslo keeps records of student results. This is done by storing a list of Prolog facts. Here is an example of facts indicating exam results:

```
res(ola,inf3110,a).
res(ola,inf2200,b).
res(ann,inf3110,c).
res(ann,inf2200,f).
res(bob,inf3110,e).
res(bob,inf2200,a).
res(max,inf3110,d).
res(max,inf2200,e).
```

The relation res(X,Y,Z) indicates that the candidate with name X has achieved the grade Z in the course Y. In this system we just store the best result so far, and do not store information about any other exam attempts.

3a

Write a query (and rules if necessary) to list all students who have taken the course inf3110.

3b

Write a query (and rules if necessary) to list all students who have *passed* the examination for inf2200, i.e. who have achieved a grade better than f in the course.

3c

A collection of res-facts is said to be OK if there is maximum one record per person per course and if all grades are either a,b,c,d,e or f.

Example: The collection of facts above would *not* be OK if we added either of the following two facts:

```
res(max,inf3110,a).
res(bob,inf3110,k).
```

Write a query (and rules) to check that a collection of res-facts are OK.

3d

This Prolog system is used when people apply for a job as teaching assistant to IFI-courses. We add two more relations: applied(X,Y) holds if and only if person X has applied for the job as teaching assistant to course Y. Example:

```
applied(max,inf3110).
applied(bob,inf3110).
applied(ann,inf2200).
```

The system also keeps record of who have been teaching assistants in previous terms, the relation ta(X,Y,Z) holds if and only if person X was teaching assistant for course Y in term Z. Example:

```
ta(max,inf3110,h04).
ta(ola,inf2200,h04).
ta(ann,inf3110,v05).
```

For a person to be eligible for the job as teaching assistant to a given course, she must have applied for the job and passed the course (i.e. achieved a grade better than f). Since teaching assistant jobs are very popular, a person may only have the job once (i.e. one term). However if no one else has applied to the given course a person may get the position even if she has been a teaching assistant for the course before.

Define the relation:

```
candidate(X,Y)
```

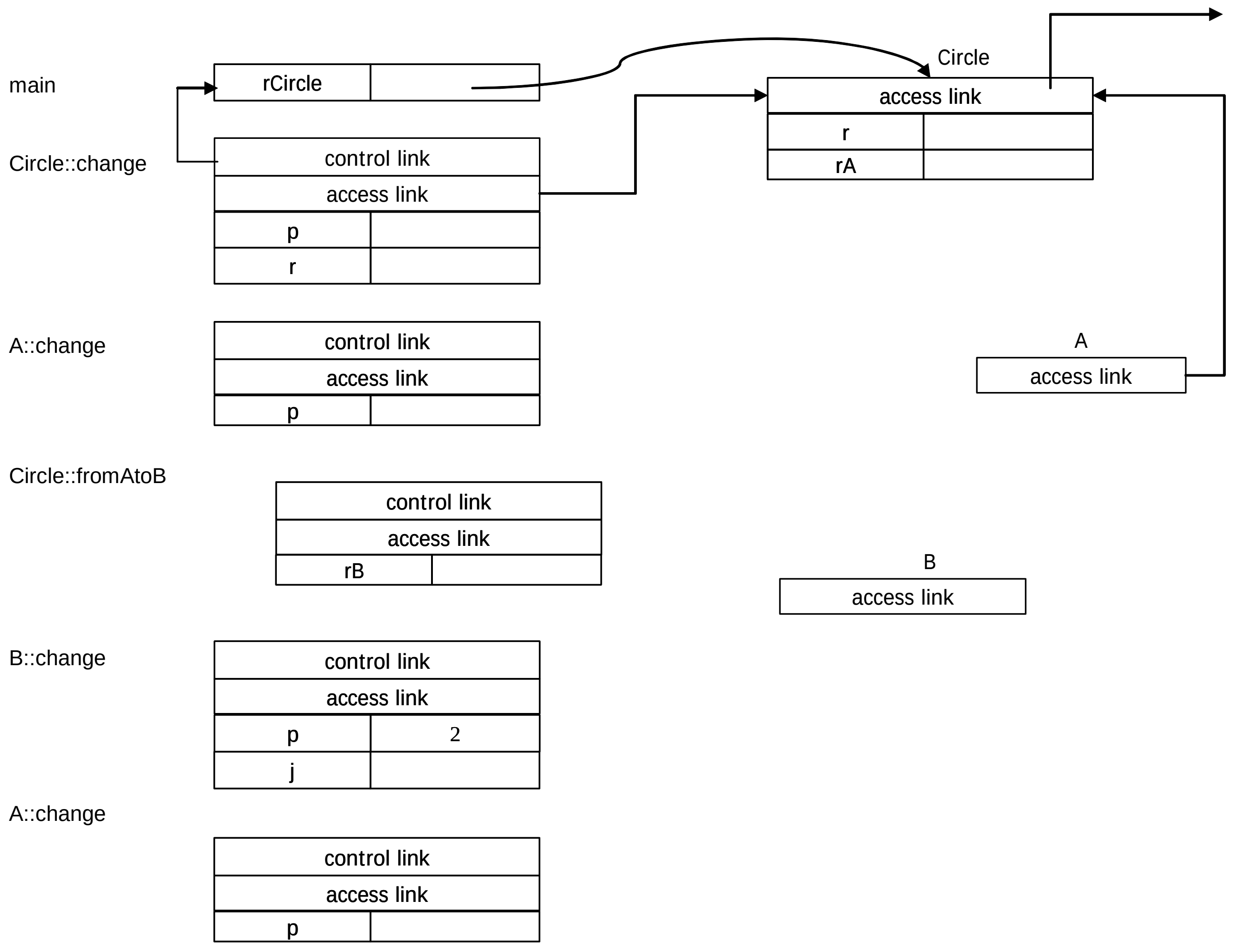
which holds if and only if X is an eligible candidate for the posistion as teaching assistant in the course Y according to the criteria given above.

Hint: You might want to define a helper relation `moreThanOneApplicant(X,Y)` which is true if and only if there is more than one applicant to the post as teaching assistant for the course Y.

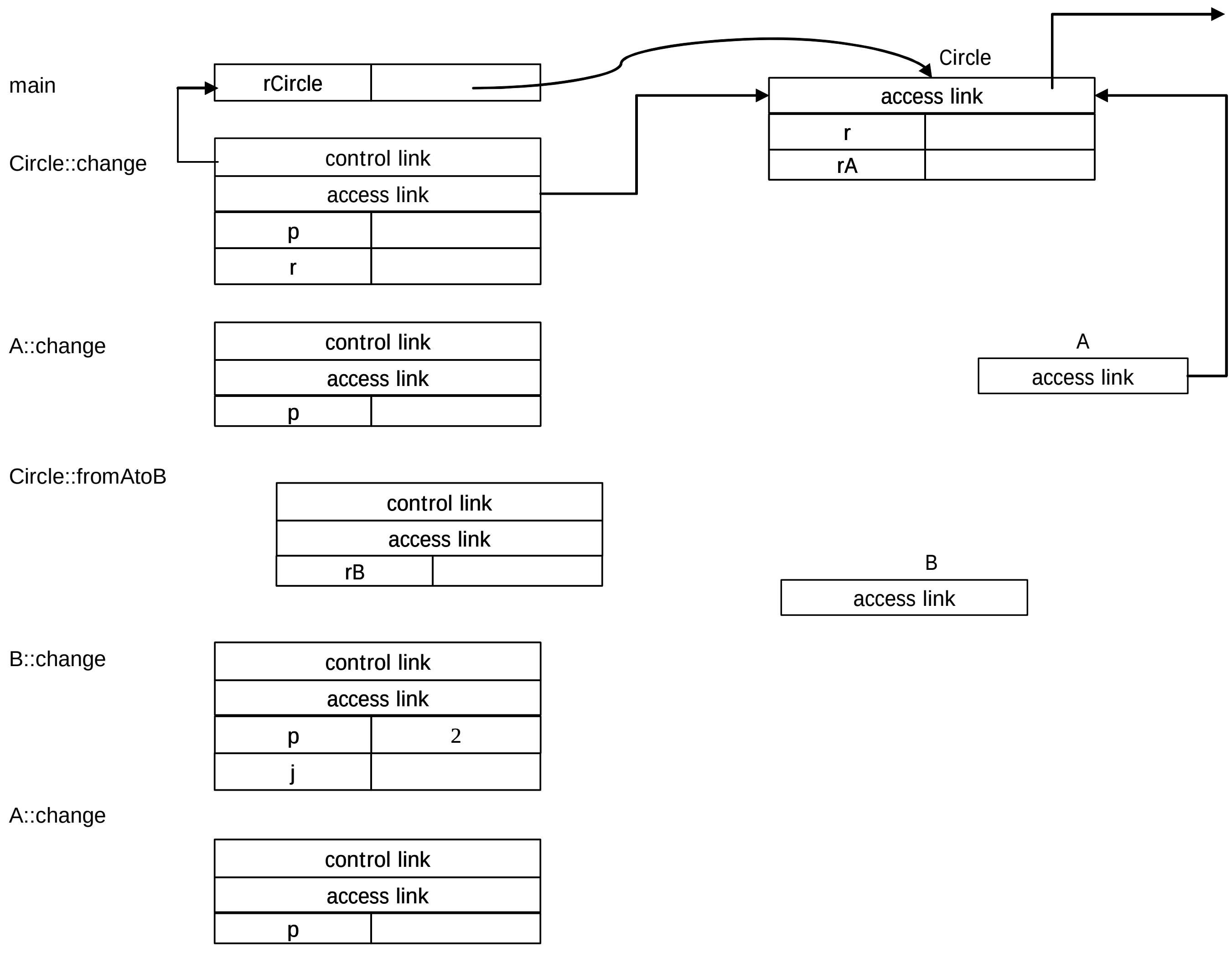
Page for answering Question 1a

Candidate no: ....

Date: ....



Extra page for sketching the answer to Question 1a


main
rCircle
Circle::change
control link
access link
p
r
A::change
control link
access link
p
Circle::fromAtoB
control link
access link
rB
B::change
control link
access link
p
2
j
A::change
control link
access link
p
Circle
access link
r
rA
A
access link
B
access link