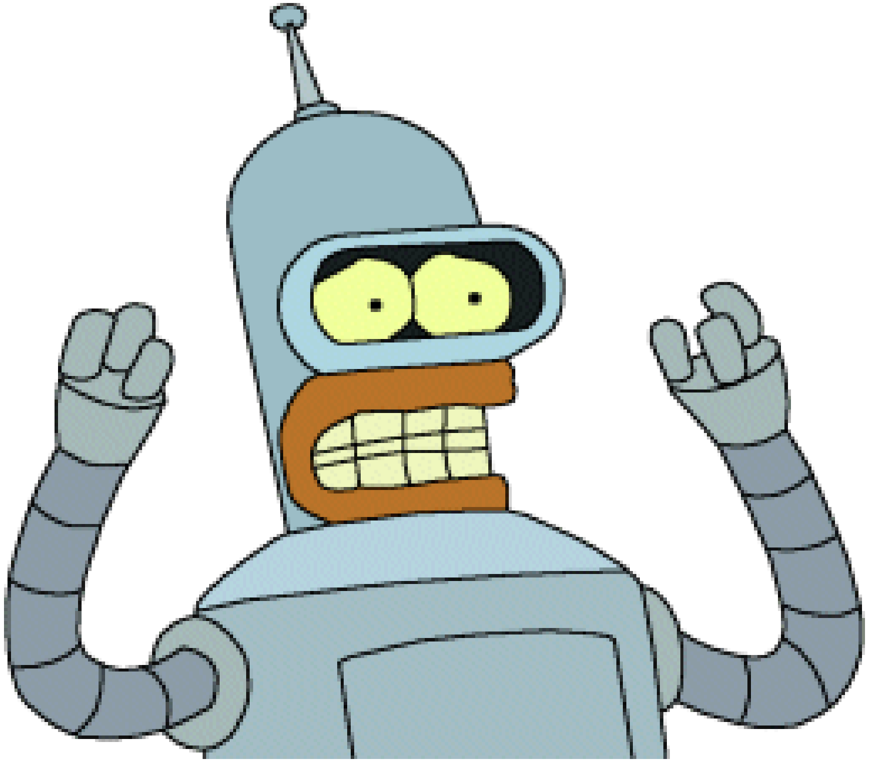
Mandatory 1 Revisited

- Make an interpreter for the ROBOL language



- Any questions?

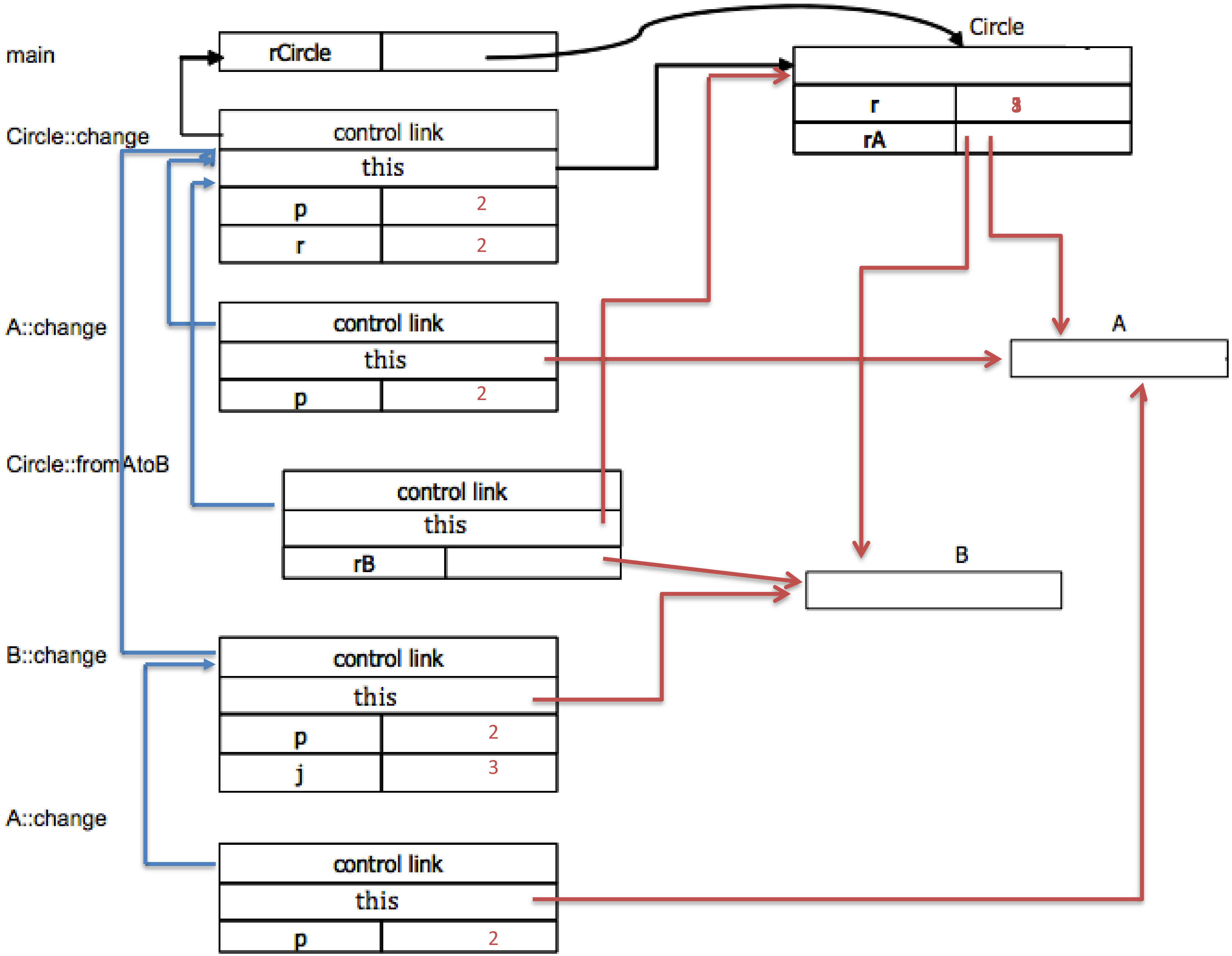
Problem 1 – Virtual methods

```
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;
    }
}
```

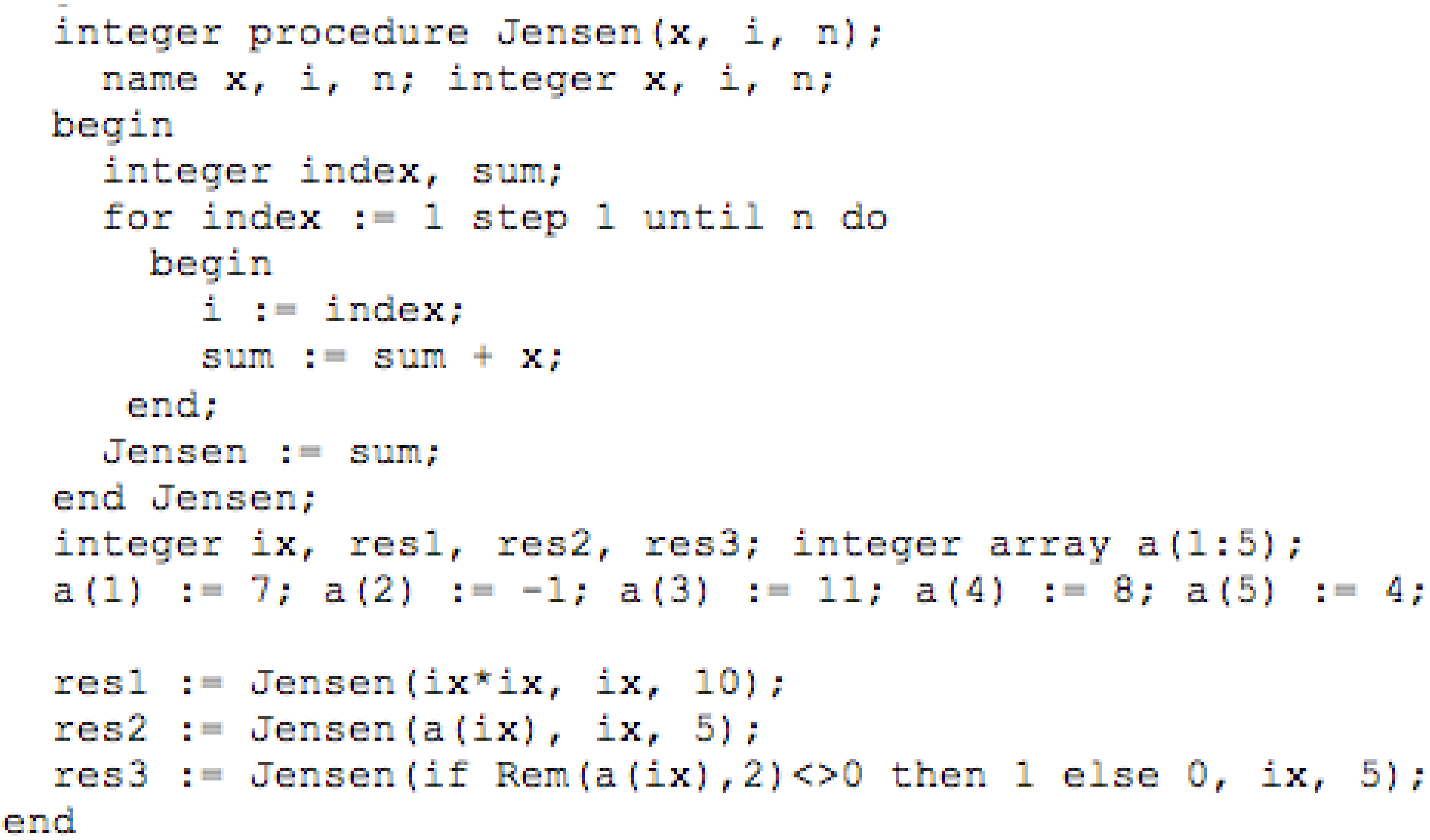
Task: Put in values, access and control links in the activation blocks (next slide) when the execution is here after the call marked (*)

Virtual method call

Polymorphism

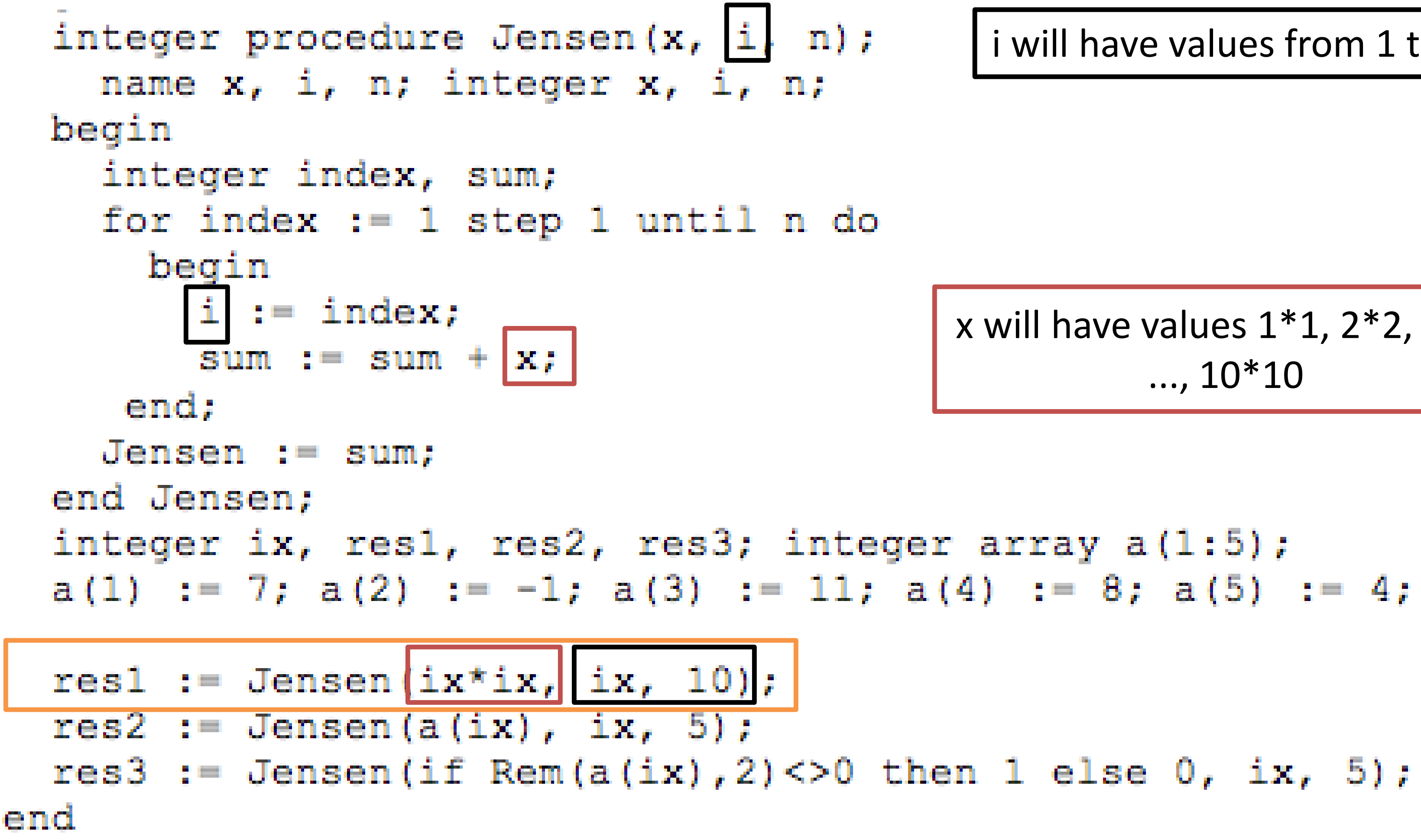

main
rCircle
Circle
Circle::change
control link
this
p
2
r
2
r
8
rA
A::change
control link
this
p
2
A
Circle::fromAtoB
control link
this
rB
B
B::change
control link
this
p
2
j
3
A::change
control link
this
p
2

Problem 2 – call by name



```
  integer procedure Jensen(x, i, n);
    name x, i, n; integer x, i, n;
  begin
    integer index, sum;
    for index := 1 step 1 until n do
      begin
        i := index;
        sum := sum + x;
     end;
    Jensen := sum;
  end Jensen;
  integer ix, res1, res2, res3; integer array a(1:5);
  a(1) := 7; a(2) := -1; a(3) := 11; a(4) := 8; a(5) := 4;

  res1 := Jensen(ix*ix, ix, 10);
  res2 := Jensen(a(ix), ix, 5);
  res3 := Jensen(if Rem(a(ix),2)<>0 then 1 else 0, ix, 5);
end
```

Problem 2 – call by name; res1



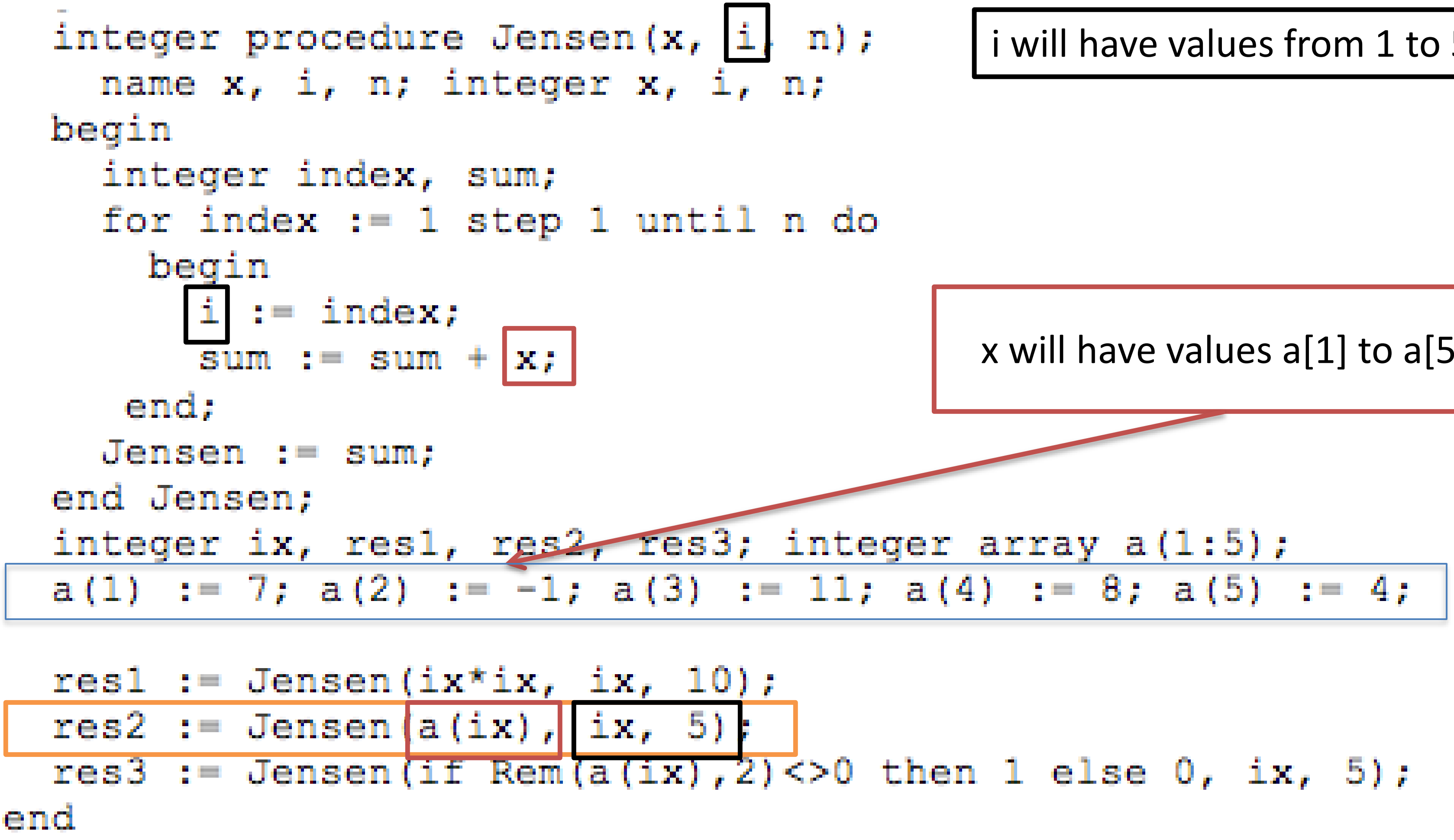
```
integer procedure Jensen(x, i, n);
  name x, i, n; integer x, i, n;
begin
  integer index, sum;
  for index := 1 step 1 until n do
    begin
      i := index;
      sum := sum + x;
    end;
  Jensen := sum;
end Jensen;
integer ix, res1, res2, res3; integer array a(1:5);
a(1) := 7; a(2) := -1; a(3) := 11; a(4) := 8; a(5) := 4;

res1 := Jensen(ix*ix, ix, 10);
res2 := Jensen(a(ix), ix, 5);
res3 := Jensen(if Rem(a(ix),2)<>0 then 1 else 0, ix, 5);
end
```

i will have values from 1 to 10

x will have values 1*1, 2*2, 3*3, ..., 10*10

res1 = 1*1 + 2*2 + 3*3 + ... 10*10

Problem 2 – call by name; res2



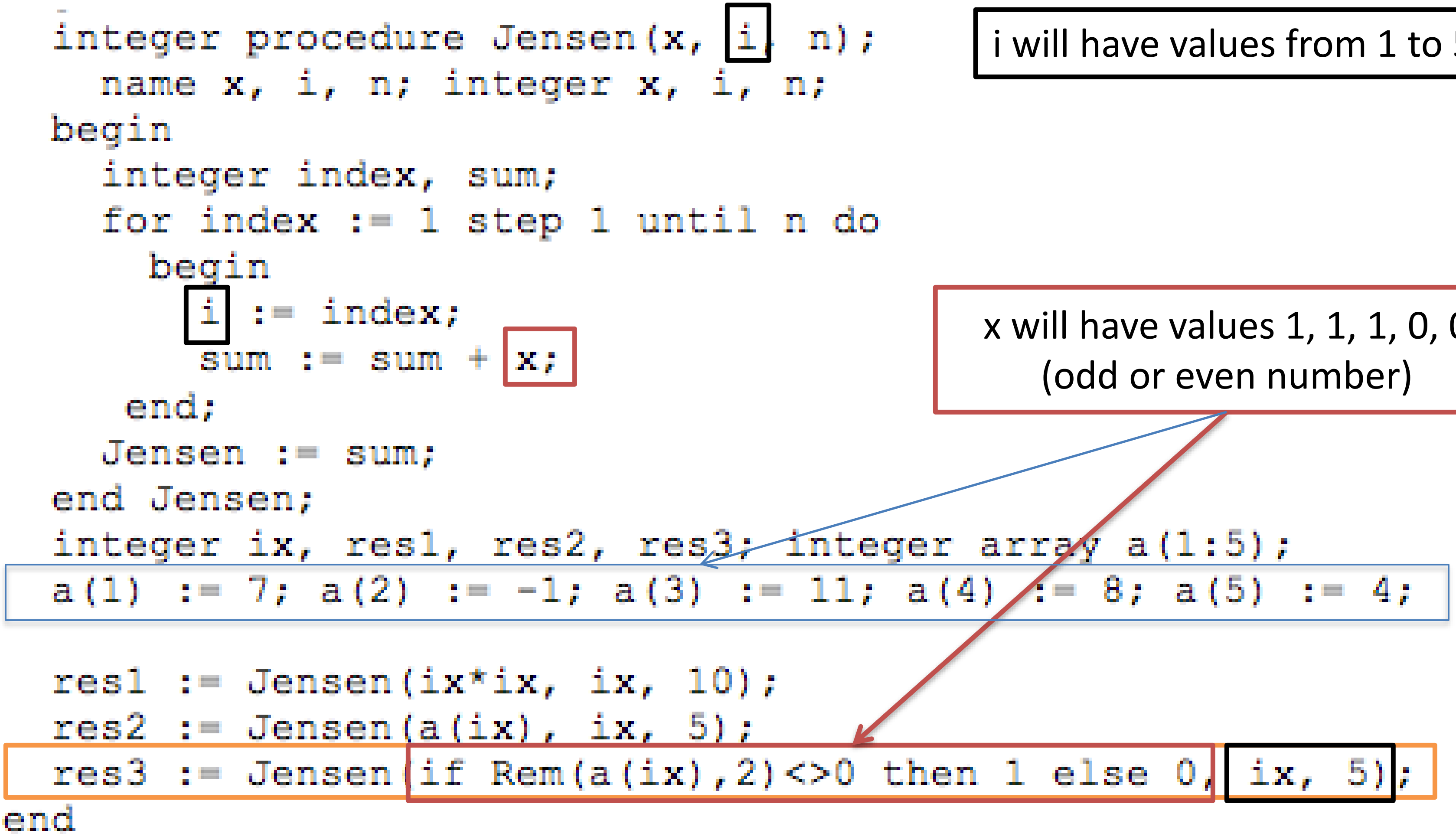
```
  integer procedure Jensen(x, i, n);
    name x, i, n; integer x, i, n;
  begin
    integer index, sum;
    for index := 1 step 1 until n do
      begin
        i := index;
        sum := sum + x;
      end;
    Jensen := sum;
  end Jensen;
  integer ix, res1, res2, res3; integer array a(1:5);
  a(1) := 7; a(2) := -1; a(3) := 11; a(4) := 8; a(5) := 4;

  res1 := Jensen(ix*ix, ix, 10);
  res2 := Jensen(a(ix), ix, 5);
  res3 := Jensen(if Rem(a(ix),2)<>0 then 1 else 0, ix, 5);
end
```

i will have values from 1 to 5

x will have values a[1] to a[5]

res2 = 7+(-1)+11+8+4

Problem 2 – call by name; res3



```
 integer procedure Jensen(x, i, n);
   name x, i, n; integer x, i, n;
 begin
   integer index, sum;
   for index := 1 step 1 until n do
     begin
       i := index;
        sum := sum + x;
    end;
   Jensen := sum;
 end Jensen;
 integer ix, res1, res2, res3; integer array a(1:5);
 a(1) := 7; a(2) := -1; a(3) := 11; a(4) := 8; a(5) := 4;

 res1 := Jensen(ix*ix, ix, 10);
 res2 := Jensen(a(ix), ix, 5);
 res3 := Jensen(if Rem(a(ix),2)<>0 then 1 else 0, ix, 5);
end
```

i will have values from 1 to 5

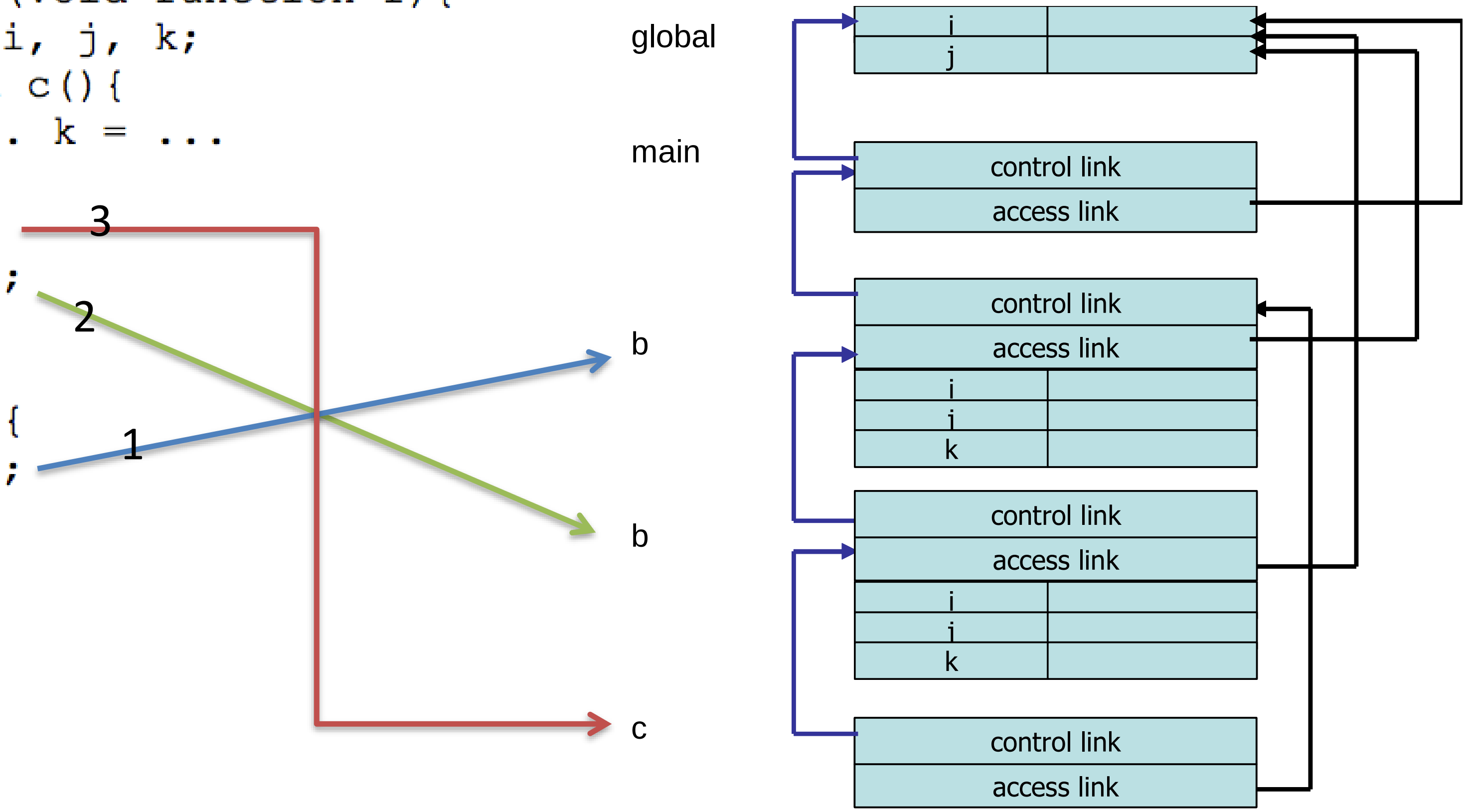
x will have values 1, 1, 1, 0, 0
(odd or even number)

res3 = 1+1+1+0+0

Problem 3 – function parameters

Draw the run-time stack as it is when f is called within b the second time.

```
{ int i, j;
  void a(){
    int k; ...
  };
  void b(void function f){
    int i, j, k;
    void c(){
      ... k = ...
     };
    f();
    b(c);
    ...
  }
  main(){
    b(a);
  }
}
```

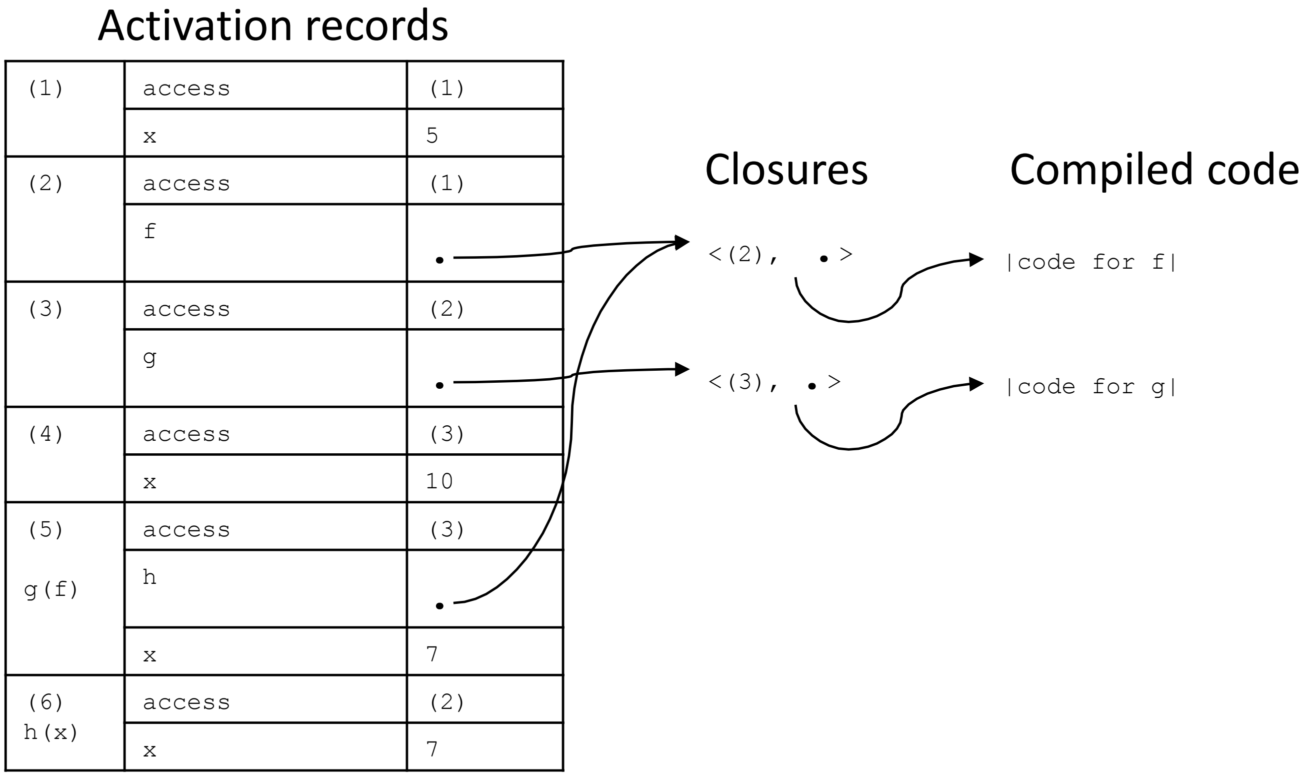


Problem 4 – Scope in ML

Fill in the missing information in the following illustration of the run-time stack after the call to h inside the body of g. Remember that function values are represented by closures and that a closure is a pair consisting of an environment (pointer to an activation record) and compiled code.

```
val x = 5;
fun f(y) = (x+y) -2;
fun g(h) = let val x = 7 in h(x) end;
let val x = 10 in g(f) end;
```

result = 5+7-2 = 10



Activation records

(1)	access	(1)
	x	5
(2)	access	(1)
	f	•
(3)	access	(2)
	g	•
(4)	access	(3)
	x	10
(5) g(f)	access	(3)
	h	•
	x	7
(6) h(x)	access	(2)
	x	7

Closures

<(2), • >

<(3), • >

Compiled code

|code for f|

|code for g|

Problem 5

- Can the L-value of a variable be accessed *only* when its name is visible (i.e. within scope)? If YES, why, and if NO, why and how?

- NO! For instance reference parameters, pointers, closures etc. Example:

```
{     -- block that does not contain i
      void f(ref int j) {    ... j= … }
  …
  {
    int i;
    f(i)
  }
}
```

Problem 6 - Determinism

- Parameters to procedures are often used in order to parameterize the computation, so that procedures called with different actual values perform different computations.
  - In which cases will a procedure without parameters not perform the same computation every time it is called?

- For instance
  - When it reads an external value (network, keyboard, pseudorandom generator, etc)
  - When it uses global variables
  - When it uses undefined operations in the language (e.g. in C)
  - Etc

Problem 7 - Call by ref vs value-result

- By-reference and by-value-result have in most cases the same effect. Consider this small example:

```
int x;
void p(int i) {
    i=i+1;
    x=x+1;
};
x=1;
p(x);
```

Will the call p(x) have the same or different effect when the parameter i is by-reference and by-value-result?

Call by ref: x = 1 +1 = 2; x = 2 + 1 = 3;
Call by value-result: i = 1 + 1 = 2; x = 1 + 1 = 2; x = 2;

Problem 8 – Functions vs call by name

- It was indicated at the lecture that functions as parameters and name parameters are similar in that the actual parameters have to maintain their environment.
- Indicate a way in which some of the properties of name parameters can be achieved by means of functions as parameters. Which property cannot be achieved in this way?
  - When the name parameter is an expression that is just used for its R-value, then a function will work in the same way
  - When the name parameter is assigned to, this will (obviously) not work.

Call by name

Variable evaluated every time it is used

What does this code give when using by-value and by-name?

```
int i = 10;

void f(int a) {
   for(...) {
        i = i + a;
    }
}
f(i);
```

By value: value of i is evaluated when calling the function. Giving us i = i + 10;

By name: nothing is evaluated when calling the function. a is evaluated every time it is used. Giving us the current value of i. Giving us i = i + i;

Call by need

Variables evaluated first time only

Gives the same value back every following use

Problem 9 – Parameters in Java

- a) Java does not have call-by-reference parameters, while C# has. How would you in Java get the effect of p(a), where a is a variable and the formal parameter is a call-by-reference parameter?
  - a = p(a); BUT, only for single-threaded programs!
- b) Java does not have call by value result parameters. How would you in Java get the effect of p(a), where a is a variable and the formal parameter is a call-by-value-result parameter.
  - a = p(a)
- c) What about p(a,b), where both are call-by-value-result parameters?
  - You create an object with values for a and b, and pass this in.