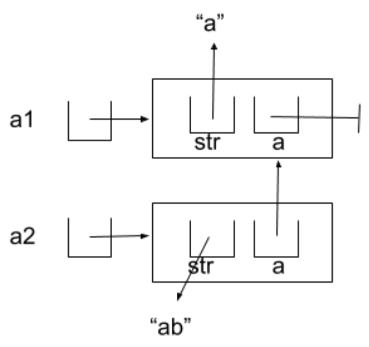
```
/*
 *   Suggested solutions exam DAT043 2020-06-10
 *
 */

    // -------------------------- 1 -------------------- (4p)

    public static void main(String[] args) {
        Scanner sc = new Scanner(in);
        out.println("Talet n > ");
        int n = sc.nextInt();
        int nCubes = getNCubes(n);
        out.println("Antal: " + nCubes);
    }

    int getNCubes(int n) {
        int c = 0;
        for (int i = 1; i <= n; i++) {
            c += i * i * i;
        }
        return c;
    }

    // -------------------------- 2 --------------------

    // No such question
// ---------------- 3 ---------------------- (7p)

   // No nested loops for full points
  int[] leaders(int arr[]) {
     int[] leaders = new int[arr.length];
     int last = arr.length - 1;
     leaders[last] = arr[last];
     int max = leaders[last];
     for (int i = arr.length - 2; i >= 0; i--) {
        if (arr[i] > max) {
           last--;
           leaders[last] = arr[i];
           max = arr[i];
        }
     }
     int c = 0;            // Remove leading zeros
     while (leaders[c] == 0) {
        c++;
     }
     int[] result = new int[leaders.length - c];
     for (int i = 0; i < result.length; i++) {
        result[i] = leaders[i + c];
     }
     return result;
  }
```

```
// ---------------- 4 --------------------- (10p)

    boolean hasSubmatrixWithSum(int[][] matrix, int sum) {
        for (int i = 1; i <= matrix.length; i++) {
            if (hasSubmatrixSizeWithSum(matrix, i, sum)) {
                return true;
            }
        }
        return false;
    }

    boolean hasSubmatrixSizeWithSum(int[][] matrix, int subSize, int sum) {
        for (int row = 0; row < matrix.length - subSize + 1; row++) {
            for (int col = 0; col < matrix.length - subSize + 1; col++) {
                if (sumMatrix(matrix, row, col, subSize) == sum) {
                    return true;
                }
            }
        }
        return false;
    }

    // Sum all elements in matrix m
    int sumMatrix(int[][] m, int row, int col, int size) {
        int sum = 0;
        for (int r = row; r < row + size; r++) {
            for (int c = col; c < col + size; c++) {
                sum = sum + m[r][c];
            }
        }
        return sum;
    }
```

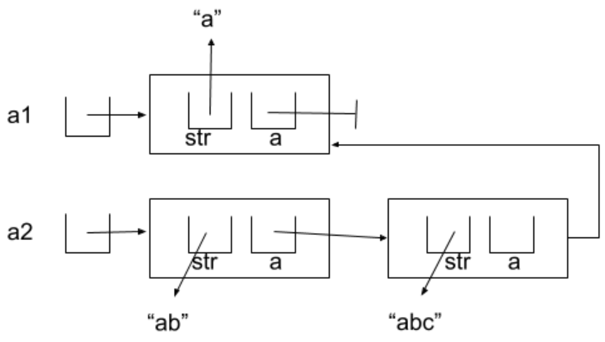
```
// ------------------------ 5 ------------------ (8p)

    String toSoundex(String name) {
        char first = name.charAt(0);
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < name.length() - 1; i++) {
            char val = getValueForChar(name.charAt(i));
            if (val != '0' && val != getValueForChar(name.charAt(i + 1))) {
                sb.append(val);
            }
        }
        sb.append(getValueForChar(name.charAt(name.length() - 1)));

        String tmp = sb.toString();
        if (getValueForChar(first) == tmp.charAt(0)) {
            tmp = tmp.substring(1);
        }
        if (tmp.length() < 3) {
            tmp = tmp + "000";
        }
        if (tmp.length() > 3) {
            tmp = tmp.substring(0, 3);
        }
        return first + tmp;
    }

     char getValueForChar(char ch) {
        String keys = "abcdefghijklmnopqrstuvwxyz";
        String values = "01230120022455012623010202";
        return values.charAt(keys.indexOf(ch));
    }
```

// ---------------- 6 ------------------------------- (8p)

Före



Efter



```
// -------------------- 7 ------------------------- (10p)

    public class Location {
        private String address;
        private final List<Trailer> trailers = new ArrayList<>();

        public Location(String address) {
            this.address = address;
        }

        public Trailer find(){
            for( Trailer t : trailers){
                if( t.getStatus() == Status.FREE){
                    return t;
                }
            }
            return null;
        }
    }
    public class Booking {
        private final Trailer trailer;
        private final Customer customer;
        private final Location pickup;
        private final Location leave;
        private final Period period;

        public Booking(Trailer trailer, Customer customer, Location pickup, Location leave,
Period period) {
            this.trailer = trailer;
            this.customer = customer;
            this.pickup = pickup;
            this.leave = leave;
            this.period = period;
        }
    }


    public class TrailerRental {
        private final List<Location> locations = new ArrayList<>();
        private final List<Booking> bookings = new ArrayList<>();

        boolean bookTrailer( Location from, Location to, Customer customer, Period period){
            if( !locations.contains(from) || !locations.contains(to)){
                return false;
            }
            Trailer t = from.find();
            if( from.find() == null){
                return false;
        }
        t.setStatus(Status.OCCUPIED);
        Booking b = new Booking(t, customer, from, to, period);
        bookings.add(b);

        return true;
    }

}
```

```
// --------------------- 8 --------------------------- (8p)

  public int lengthOfLIS(int[] arr) {
     if (arr == null || arr.length == 0)
        return 0;

     // Use for book keeping
     int[] len = new int[arr.length];
     Arrays.fill(len, 1);

     // Keep track of max in len array (avoid another loop)
     int maxLen = 1;

     // For all elements from 1 ...
     for (int i = 1; i < arr.length; i++) {
        // For all to the left of arr[i] in turn
        for (int j = 0; j < i; j++) {
           // If arr[i] > arr[j] then j to i is a (short) sequence
           // len[j] holding length to j,
           // so length to i should be len[j] + 1.
           // But there could be other longer sequences to i
           // must check to see which is longest, the new formed
           // with j and i or the existing one.
           // Example: 4,5,1,7
           if (arr[i] > arr[j]) {
              // Update len with longest
              len[i] = Math.max(len[i], len[j] + 1);
           }
        }
        // Update maxLen if len[i] is new max
        maxLen = Math.max(maxLen, len[i]);
     }
     return maxLen;
  }
```