1 1. (25%)

Se på følgende metode som tester om det finnes et trippel som har sum mindre enn 0 i en tabell av heltall:

```
boolean existsThreeSumNeg(int[] t) {
  int N = t.length;
  for (int i = 0; i < N-2; i++)
    for (int j = i+1; j < N-1; j++)
      for (int k = j+1; k < N; k++)
      if ( t[i] + t[j] + t[k] < 0 ) { return true; }
  return false;
}
```

a) Hva er kjøretid i verste fall til existsThreeSumNeg(t) ? Grunngi svaret ditt.

b) Skriv en metode existsThreeSumNeg(int[] t) som er mer effektiv i det verste tilfellet enn metoden over. For å få alle poeng skal din metode kjøre i O(N) tid (ja, det er mulig!).

c) Nå antar vi at tabellen t inneholder heltall som med 50% sannsynlighet er negative. Er det da mulig å skrive en metode existsThreeSumNeg(int[] t) som har kjøretid som er i snitt konstant, det vil si, O(1)? Grunngi svaret ditt.

Skriv ditt svar her...

```
1
```

Maks poeng: 25

2 2. (25%)

Anta at følgende metode er gitt:

```
// precondition: 0<=lo<=mid<hi<=a.length, and a[lo..mid] and a[mid+1..hi] are sorted
void merge(Comparable[] a, int lo, int mid, int hi)
// postcondition: a[lo..hi] is sorted
```

a) Skriv en metode (some bruker metoden merge() over, og ingen andre)

```
void sort(Comparable[] a, int lo, int hi)
```

som sorterer a[lo..hi].

b) Hva er kjøretid i verste fall til din metode sort(), gitt at metoden merge() kjører i tid O(hi-lo)? Forklar svaret ditt i tilfelle N = a.length er en potens av 2.

Skriv ditt svar her...

1

Maks poeng: 25

3 **3. (25%)**

Se på følgende ufullstendige program for en hash-tabell med separate chaining.

```
//simple symbol table based on hashing with separate chaining
public class SeparateChainingHashST<Key, Value> {

  private ArrayList<KeyValueNext> table; // the actual hash table
  private int M;     // length of the hash table (not the number of keys)
  private class KeyValueNext {
     private Key key;
     private Value value;
     private KeyValueNext next;
     public KeyValueNext(Key key, Value val, KeyValueNext next){
        this.key = key; this.value = val; this.next = next; }
  }

public SeparateChainingHashST() {
    M = 31; // typical values of M are 31, 997, 65521, 1048573
    table  = new ArrayList<KeyValueNext>(M);
    for(int i=0; i<M; i++){ table.add(null); }
}

private int hash(Key key){
    return ( key.hashCode() & 0x7fffffff ) % M;
}

public Value get(Key key) { // returns value if key is present, null otherwise
    KeyValueNext p = table.get(hash(key));
     <<< insert answer for (a) here >>
}
```

```
        public void put(Key key, Value val) { // updates value of key if key is present,
                                              // adds pair (key,val) if key is not present
            int hash = hash(key);
            KeyValueNext p = table.get(hash);
            while (p !=  null && !p.key.equals(key)) {
               p = p.next;
            }
              <<< insert answer for (b) here >>}
        }

        public static void main(String[] args)  {
            In infile = new In(args[0]);
            SeparateChainingHashST<String,Integer> st = new SeparateChainingHashST<>();
            while (!infile.isEmpty()) {
               String key = infile.readString();
        <<< insert answer for (c) here >>
}//End of main
}// End of class
```

a) Skriv ferdig metoden getpos(Key key).

b) Skriv ferdig metoden put(Key key, Value v). Du trenger å bruke metoden set(_,_) til ArrayList.

c) Skriv ferdig metoden main(String[] args) slik at tabellen inneholder parene (streng,frekvens) for enhver streng som forekommer i input-filen, der frekvens er hvor ofte den forekommer.

Skriv ditt svar her...

```
1
```

Maks poeng: 25

4 4. (25%)

Se på følgende program for å finne stier i en labyrinth.

```
import edu.princeton.cs.algs4.StdOut;

// simple pathfinder in maze
public class FunnyMaze {
private static boolean searchPath(// searches a path in the maze
// from (inx,iny) to (outx,outx) via cells that are true
// by horizontal, vertical and diagonal steps
```

```
            int N, // size of the square maze NxN
            int inx, int iny, int outx, int outy,
            boolean[][] maze, boolean[][] visited, boolean[][] onpath) {
            if (!maze[inx][iny] || visited[inx][iny] ) {return false;}
            onpath[inx][iny] = true;
            StdOut.println("at " + inx + "," + iny);                  // print statement 1
            if (inx == outx && iny == outy ) {return true;}
            visited[inx][iny] = true;
            for (int x = Math.max(inx-1,0); x < Math.min(inx+2,N); ++x)
               for (int y = Math.max(iny-1,0); y < Math.min(iny+2,N); ++y){
                  if (!visited[x][y] &&
                      searchPath(N,x,y,outx,outy,maze,visited,onpath))
                     { return true; }
               }
            onpath[inx][iny] = false;
            StdOut.println("not on path " + inx + "," + iny);  // print statement 2
            return false;
         }
         public static void main(String[] args)  {
            boolean[][] onpath = new boolean[4][4]; // all cells initially false
            boolean[][] visited = new boolean[4][4]; // all cells initially false
            boolean[][] maze = new boolean[4][];
            maze[0] = new boolean[]{true,true,false,true};
            maze[1] = new boolean[]{true,false,false,false};
            maze[2] = new boolean[]{true,true,false,true};
            maze[3] = new boolean[]{true,false,true,true};

            searchPath(4,0,0,3,3,maze,visited,onpath);
            for (int x=0; x<4; ++x){                    // print loop
               for(int y=0; y<4; ++y)
                  if (onpath[x][y]) {StdOut.print(1);}
                  else {StdOut.print(0);}
               StdOut.println();                        // end of print loop
             }
         } // end of main
         } // end of class FunnyMaze
```

a) Hva blir skrevet ut når programmet over blir kjørt?

b) Finner programmet over alltid en sti som er kortest? Hvis ja, forklar svaret ditt.

Hvis nei, skisser en annen algoritme som alltid finner en sti som er kortest (hvis de to punktene er forbundet i det hele tatt).

Skriv ditt svar her...

```
1
```

Maks poeng: 25