1. Tempo disponibile 120 minuti.
2. Non è possibile consultare appunti, slide, libri, persone, siti web, ecc.
3. Scrivere in modo leggibile, su ogni foglio, nome, cognome e numero di matricola.
4. Le soluzioni agli esercizi che richiedono di progettare un algoritmo devono:
   - spiegare a parole l'algoritmo (se utile, anche con l'aiuto di esempi o disegni),
   - fornire e commentare lo pseudo-codice (indicando il significato delle variabili),
   - calcolare la complessità (con tutti i passaggi matematici necessari),
   - se l'esercizio ammette più soluzioni, a soluzioni computazionalmente più efficienti e/o concettualmente più semplici sono assegnati punteggi maggiori.

1. Calcolare la complessità $T(n)$ della seguente procedura `mistery`:

```
function mistery(n: Int) --> Int
  k = n
  a = 0
  while k>1
    a = a+mistery2(n,k)
    k = k/2
  end while
  return a

function mistery2(n: Int, k: Int) --> Int
  if n>1
    return mistery2(n/2,k+1)+mistery2(n/2,k+2)
  else
    return k
```

2. Si consideri un max-heap binario `H` rappresentato in memoria dal vettore `[15,12,13,9,7,1,3,8,5,6]`. Assumete di eseguire sull'heap `H` le seguenti operazioni, una dopo l'altra:
   (a) `H.deleteMax()`
   (b) `H.insert(11)`
   (c) `H.insert(14)`
   (d) `H.deleteMax()`

   Indicare come cambia la rappresentazione in memoria dell'heap dopo l'esecuzione di ognuna di queste operazioni.

3. In una fiera di streetfood, una intera via è dedicata a venditori di hamburger. Ogni venditore propone un proprio hamburger, solitamente di peso diverso rispetto agli altri venditori. Percorriamo la strada passando davanti ai venditori in modo sequenziale, dal primo all'ultimo senza mai tornare indietro. Quando passiamo davanti ad un venditore possiamo decidere se acquistare o meno un hamburger. Se decidiamo di acquistarlo, questo deve comunque avere un peso non inferiore al peso dell'ultimo hamburger acquistato. Progettare un algoritmo che, dato il vettore `K[1..n]` con i pesi degli hamburger proposti dai vari venditori che si incontrano lungo la strada (ad esempio, `K[1]` è il peso dell'hamburger del primo venditore e `K[n]` il peso dell'hamburger dell'ultimo venditore), calcola il massimo numero di hamburger che si possono acquistare.

4. Progettare un algoritmo che, dato un grafo non orientato aciclico $G = (V, E)$, ovvero un albero libero, ed un nodo $r \in V$, calcola l'altezza del corrispondente albero radicato in r.