Nom i cognoms:

Algorísmia i Programació 1, GCED, 17 de gener de 2019

L'examen dura dues hores i mitja. Es valorarà la concisió, claredat i brevetat a més de la completesa i l'exactitud de les respostes. Contesteu a l'espai marcat amb bona lletra. Poseu el vostre nom a cada full. No podeu consultar cap material addicional.

1 Pupurri (2 punts)

1. Què vol dir el veredicte *Accepted* (semàfor verd) del Jutge?

2. Expliqueu per a què serveix un *assert*.

3. Expliqueu per a què serveix i quins requeriments té l'algorisme de fusió.

4. Doneu un avantatge de Python sobre C++.

5. Doneu un avantatge de C++ sobre Python.

6. Expliqueu quina diferència essencial hi ha entre els enters de C++ i els de Python.

7. Quin benefici dóna compilar els programes en C++ amb `-Wall`?

8. Expliqueu perquè és convenient indentar els programes.

9. Expliqueu la diferència entre una *acció* i una *funció*.

10. Què escriuen aquests programes en C++ i Python3?

```
void cdr (vector<int>& a, vector<int>& b) {
    a = b;
    for (int i = 0 ; i < a.size() ; ++i) a[i] /= 2;
}

int main () {
    vector<int> u = {3, 4, 2};
    vector<int> v = {5, 4, 1};
    cdr(u, v);
    for (int x : u) cout << x << ' ';
}
```

```
def cdr (a, b):
    a = b
    for i in range(len(a)) : a[i] /= 2


u = [3, 4, 2]
v = [5, 4, 1]
cdr(u, v)
print(u)
```

Nom i cognoms:

2 Garbell d'Eratòstenes (3 punts)

Aquesta és una implementació de l'algorisme del Garball d'Eratòstenes:

```
vector<bool> eratostenes(int n) {
    assert( ??? );
    vector<bool> p(n + 1, true);
    p[0] = p[1] = false;
    for (int i = 2 ; i*i ≤ n ; ++i) {
        if (p[i] == true) {
            for (int j = 2*i ; j ≤ n ; j += i) {
                p[j] = false;
            }
        }
    }
    return p;
}
```

1. Quina condició escriuríeu dins de l'*assert()*?

2. Doneu l'especificació d'aquesta funció (no expliqueu com funciona l'algorisme, sinó què retorna).

3. El programa conté un petit detall que demostra mal estil de programació. Quin és?

4. Doneu l'invariant del bucle extern (el que usa la i).

5. Utilitzant l'invariant del bucle extern, expliqueu per què l'algorisme retorna el resultat correcte.

Nom i cognoms:

3 Generació combinatòria (3 punts)

Donats n nombres enters positius $[a_0, a_1, \ldots, a_{n-1}]$ i dos altres nombres enters ℓ i u tals que

$$0 \leq \ell \leq u \leq s = \sum_{i=0}^{n-1} a_i,$$

es volen escriure totes les possibles solucions $[x_0, x_1, \ldots, x_{n-1}] \in \{0,1\}^n$ de la doble inequació

$$\ell \leq \sum_{i=0}^{n-1} a_i x_i \leq u.$$

en ordre lexicogràfic creixent.

Per exemple, per a l'entrada $n = 3$, $\ell = 3$, $u = 5$ i $a = [1,2,3]$, es vol obtenir aquesta sortida:

[0, 0, 1]
[0, 1, 1]
[1, 0, 1]
[1, 1, 0]

1. Completeu el programa següent en Python3 perquè resolgui el problema anterior.

```
from jutge import read

def solucions (n, ℓ, u, a, s, x, k, sum_chosen, sum_rest):
    if [                              ] :
        print(x)
    else:
        if [                              ] :
            x[k] = 0
            solucions (n, ℓ, u, a, s, x, k + 1, sum_chosen, sum_rest − a[k])
        if [                              ] :
            x[k] = 1
            solucions (n, ℓ, u, a, s, x, k + 1, sum_chosen + a[k], sum_rest − a[k])

def main():
    n, ℓ, u = read(int, int, int)
    a = [read(int) for i in range(n)]
    s = sum(a)  # calcula la suma dels elements en a
    x = [None for i in range(n)]
    solucions (n, ℓ, u, a, s, x, [      ] , [      ] , [      ] )

main()
```

2. Doneu l'especificació de la funció *solucions* (no expliqueu com funciona l'algorisme, sinó què fa).

3. Expliqueu com modificar la funció *solucions* perquè s'escriguin les solucions en ordre lexicogràfic *decreixent* (és a dir, en l'ordre invers de l'anterior) sense usar espai addicional. En particular, no es considerarà com a resposta vàlida l'algorisme consistent a guardar tota la sortida en un vector i, al final, revessar-lo abans d'escriure'l.

4. Per a una n i unes a fixades, quins valors cal donar a ℓ i u perquè l'algorisme anterior sigui el màxim de lent?

5. Utilitzeu notació asimptòtica per descriure el nombre de vegades que l'algorisme anterior crida a **print** amb els valors ℓ i u de la pregunta anterior.

Nom i cognoms:

4 Maleïda cerca binària (2 punts)

Donat un vector ordenat v amb n elements i un element x, cal retornar -1 si x no està en v o un índex i tal que $v[i] = x$ altrament.

A continuació es dónen tres funcions per resoldre el problema anterior (i de quina font s'han obtingut). Per a cada funció, esbrineu si és correcta o no. Si ho és, justifiqueu perquè. Si no ho és, doneu un breu contraexample que demostri el seu mal funcionament.

1. Funció 1 (trobada en una web per a programadors).

```
int cercaBinaria1 (vector<int>& T, int x) {
    int l = 0;
    int r = T.size() − 1;
    while (l ≤ r) {
        int m = (l+r) / 2;
        if (x < T[m])  r = m;
        else if (x > T[m])  l = m + 1;
        else return m;
    }
    return −1;
}
```

2. Funció 2 (traducció a C++ del codi del llibre *Modern software development using Java* de Tymann i Schneider).

```
int cercaBinaria2 (vector<int>& array, int target) {
    int start = 0;
    int end = array.size();
    int position = −1;
    while (start≤end and position==−1) {
        int middle = (start + end) / 2;
        if (target < array[middle]) end = middle − 1;
        else if (target > array[middle]) start = middle + 1;
        else position = middle;
    }
    return position;
}
```

3. Funció 3 (traducció a C++ del codi del llibre *Developing Java software* de Winder i Roberts).

```
int cercaBinaria3 (vector<int>& v, int o) {
    int hi = v.size();
    int lo = 0;
    while (true) {
        int centre = (hi+lo)/2;
        if (centre == lo) {
            if (v[centre] == o) return centre;
            else if (v[centre+1] == o) return centre + 1;
            else return −1;
        }
        if (v[centre] < o) lo = centre;
        else if (o < v[centre]) hi = centre;
        else return centre;
    }
}
```