Nom i cognoms:

Algorísmia i Programació 1, GCED, 8 de gener de 2021

L'examen dura tres hores. 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 a l'espai indicat. No podeu consultar cap material addicional. Tots els problemes compten 2 punts.

1 Les preguntetes d'en Jordi

1. Què és un *algorisme*?

2. Expliqueu molt breument la diferència entre un *compilador* i un *intèrpret*.

3. Expliqueu què és una *precondició*.

4. Anomeneu quatre algorismes d'ordenació diferents vistos al curs d'AP1. Subratlleu el(s) més ràpid(s) de tots ells (en el cas pitjor).

5. Expliqueu perquè dins de la notació asimptòtica no es precisa la base dels logaritmes.

6. Digueu, tot utilitzant notació asimptòtica, quina és l'eficiència de l'algorisme de multiplicació que us van ensenyar a l'escola per multiplicar dos naturals d'n dígits.

7. La Marta ha mesurat que, al seu ordinador, l'algorisme d'ordenació per selecció triga uns 15 segons a ordenar 200000 elements. Doneu una predicció aproximada pel temps necessari per ordenar 400000 elements.

8. L'Àngel ha mesurat que, al seu ordinador, l'algorisme de cerca binària triga un segon per comprovar si un element es troba o no en un vector ordenat d'un milió elements. Què en penseu?

9. Digueu què escriuen aquests programes en Python i C++:

```
void poop (string& a, string& b) {
    string c = a;
    a = b;
    b = c;
}

int main () {
    string x = "rainbow";
    string y = "unicorn";
    poop (x, y);
    cout << x << " " << y << endl;
}
```

```
def poop (a, b):
    c = a
    a = b
    b = c


x = "rainbow"
y = "unicorn"
poop (x, y)
print (x, y)
```

Nom i cognoms:

2 Trobeu els errors

Aquests cins fragments de programes (escrits per companys vostres) contenen alguns errors de programació i algunes millores d'estíl. Expliqueu quins són i com arreglar-los (escriviu-ho directament al costat del codi). Nota: Ignoreu les qüestions de format (indentació, espaiat...).

```
def perimetre_cercle (radi):
    return 2 * 3.1416 * radi
```

```
// Donat un vector ordenat (excepte, potser, el darrer element), el deixa tot ordenat.
void inserta (vector<double>& v) {
    int posicio = v.size() - 1;
    int darrer = v[posicio];
    while (v[posicio - 1] > darrer and posicio - 1 >= 0) {
        v[posicio] = v[posicio - 1];
        --posicio;
    }
    v[posicio] = darrer;
}
```

```
// Calcula la suma de dues matrius rectangulars.
Matriu suma (const Matriu& a, const Matriu& b) {
    int n = a.size();
    int m = a[0].size();
    assert(b.size() == n and b[0].size() == m);
    Matriu c(n, Fila(m));
    for (int i = 0; i < n; ++i) for (int j = 0; j < n; ++j) c[i][j] = a[i][k] + b[i][j];
    return c;
}
```

```
# suma un segon a l'hora h:m:s.
def suma_un_segon (h, m, s):
    s += 1
    if s == 60:
        s = 0
        m += 1
        if m == 60:
            m = 0
            h += 1
            if h == 24:
                h = 0

h, m, s = 23, 59, 59
suma_un_segon(h, m, s)
print(h, m, s)  # hauria d'escriure 0 0 0
```

```
// Programa que filtra els nombres primers d` una sequencia
// tot utilitzant l` algorisme d` Eratostenes.

void eratostenes (vector<bool>& v) {
    v[0] = false;  v[1] = false;
    for (int i = 2; i * i ≤ 1000000; ++i) {
        if (v[i] == true) {
            for (int j = 2 * i; j ≤ 1000000; j += i) {
                v[j] = false;
            }
        }
    }
}

int main() {
    int n;
    vector<bool> v(1000000, true);
    eratostenes(v);
    while (cin ≫ n) {
        if (v[n] == true) cout ≪ n ≪ endl;
    }
}
```

Nom i cognoms:

3 Especificació de funcions misterioses

Considereu aquest fragment de programa en Python i especifiqueu què calculen les quatres funcions misterioses:

```
def rec (n, t, f ):
    if n == 0:  return t
    else:  return f (n , rec (n − 1, t, f ) )

def mul (a, b):  return a ∗ b

def add (a, b):  return a + b

def nxt (a, b):  return a + 1

def rnd (a, b):
    x = random.random()  # nombre real aleatori entre 0 i 1
    y = random.random()  # nombre real aleatori entre 0 i 1
    if math.sqrt(x∗x + y∗y) ≤ 1:  return b + 1
    else:  return b

def mystery1 (n):  return rec(n, 1, mul)

def mystery2 (n):  return rec(n, 0, add)

def mystery3 (n):  return rec(n, 1, nxt)

def mystery4 (n):  return rec(n, 0, rnd) / n ∗ 4
```

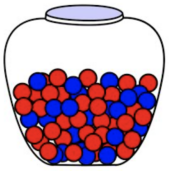
mystery1:

mystery2:

mystery3:

mystery4:

4 El pot de bales



Tenim un pot amb $n \geq 1$ bales. Cada bala pot ser blava o roja. Executem l'algorisme següent:

Entrada: Un pot amb $n \geq 1$ bales blaves i rojes.
Sortida : Un pot amb bales.

bool p = el nombre inicial de bales blaves al pot és parell

int $i = n$

while $i \neq 1$ **do**

    es treuen dues bales arbitràries del pot

    if les dues bales són del mateix color **then**

        es posa una nova bala roja al pot

    else

        es posa una nova bala blava al pot

    end

    $i = i - 1$

end

(a) Doneu un invariant (no gràfic) pel bucle de l'algorisme.

Pista: Useu b i r per referir-vos al nombre actual de bales blaves i rojes al pot i relacioneu aquests valors amb i i p.

(b) Demostreu que el vostre invariant és correcte.

(c) Utilitzeu el vostre invariant per demostrar aquests dos fets:

- Si el pot contenia un nombre parell de bales blaves a l'inici, llavors el pot conté una sola bala roja al finalitzar l'algorisme.
- Si el pot contenia un nombre senar de bales blaves a l'inici, llavors el pot conté una sola bala blava al finalitzar l'algorisme.

Nom i cognoms:

5 Quatre en ratlla

Volem dissenyar un programa que permeti jugar al joc del Quatre en ratlla. A la nostra versió del joc, hi ha n jugadors que juguen per torns en un tauler de c columnes tot col·locant fitxes del seu color per intentar ser el primer a aconseguir posar quatre fitxes seguides en ratlla (horitzontalment, verticalment o en diagonal). El tauler consta d'un nombre arbitràriament gran de files (tantes com calgui). A cada jugada, el jugador tria una columna i posa la seva fitxa damunt de totes les fitxes que aquesta ja pugui contenir. Les tirades invalides fan perdre el torn al jugador corresponent. El nombre i noms dels jugadors es dóna al principi, igual que el nombre de columnes. A continuació, es donen les tirades dels jugadors. El programa ha de donar el nom del primer jugador que forma un 4 en ratlla.

Aquest és un exemple d'interacció amb el joc:

```
3 Anna Berta Carla                  # nombre o noms dels jugadors
6                                   # nombre de columnes

3                                   # juga Anna
3                                   # juga Berta
8                                   # juga Carla (jugada invalida)
4                                   # juga Anna
4                                   # juga Berta
2                                   # juga Carla
6                                   # juga Anna
3                                   # juga Berta
1                                   # juga Carla
5                                   # juga Anna (i guanya)
```

En aquest cas, el programa ha d'anunciar que la Berta ha guanyat i que el tauler final és

```
. . B . . .
. C B B . .
C C A A A A
```

No us espanteu: En aquest problema no heu de construir tot un programa per aquesta tasca, sinó mostrar com l'organitzaríeu i estructuraríeu en C++. Per a fer-ho:

- Definiu els tipus de dades necessaris per representar una partida. Procureu que els identificadors siguin tant explicatius com pugueu i afegiu una petita documentació si us cal descriure amb més detall el seu ús. Mostreu com representaríeu l'estat del joc al final de l'exemple anterior.
- Definiu les capçaleres de les operacions (funcions i accions) principals que considereu que són essencials per implementar el joc de forma estructurada, tot utilitzant com a paràmetres valors de tipus bàsics i dels tipus definits anteriorment. Especifiqueu amb precisió però amb consició cadascuna d'aquestes operacions.

Fixeu-vos que no heu de donar codi més enllà de les declaracions de tipus i operacions.

Nom i cognoms:

Nom i cognoms: