Cognoms **Nom** **DNI**

Examen Parcial EDA **Duració: 1h 30min** **15/04/2021**

- *L'enunciat té 4 fulls, 8 cares, i 2 problemes.*
- *Poseu el vostre nom complet i número de DNI a cada problema.*
- *Contesteu tots els problemes en el propi full de l'enunciat a l'espai reservat.*
- *Llevat que es digui el contrari, sempre que parlem de cost ens referim a cost asimptòtic en temps.*
- *Llevat que es digui el contrari,* **cal justificar les respostes.**

Problema 1 **(5 pts.)**

Responeu a les següents preguntes:

(a) (1.25 pts.) Escriviu C o F dins de cada casella indicant si l'afirmació corresponent és certa o falsa, respectivament:

$2^{2n} \in O(2^n)$ ☐	$\log(2n) \in O(\log(n))$ ☐
$2^{2n} \in \Omega(2^n)$ ☐	$\log(2n) \in \Omega(\log(n))$ ☐
$2^{2n} \in \Theta(2^n)$ ☐	$\log(2n) \in \Theta(\log(n))$ ☐

Justifiqueu la vostra resposta:

(b) (1.25 pts.) Considereu el codi següent:

```
int x = 2;
int y = 1;
while (y ≤ n) {
  y = y + x;
  x = x + 1;
}
```

En funció de n, el seu cost és $\Theta($ [ ] $)$. Justifiqueu la vostra resposta:

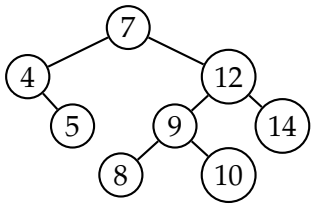
(c) (1.25 pts). Considereu el codi següent:

```
bool f(const map<int,int>& M, const vector<int>& v) {
  for (int x : v)
    if (M.find(x) ≠ M.end()) return true;
  return false;
}


int main() {
  int n; cin ≫ n;

  map<int,int> M;
  for (int i = 0; i < n; ++i) {
    int x; cin ≫ x;
    ++M[x];
  }

  vector<int> v(n);
  for (int i = 0; i < n; ++i) cin ≫ v[i];

  cout ≪ f(M,v) ≪ endl;
}
```

Cognoms **Nom** **DNI**

Què fa el codi anterior? Quin és el cost en cas pitjor d'una crida a f en funció d'n?

(d) (1.25 pts.) Escriviu l'arbre AVL resultant d'afegir l'element amb clau 11 a l'arbre AVL següent. No cal justificar la resposta.

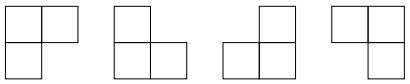


Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.

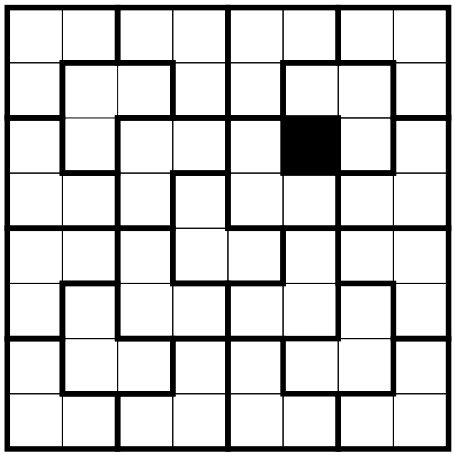
Cognoms **Nom** **DNI**

Problema 2 **(5 pts.)**

Donada una graella de mida $2^n \times 2^n$, amb $n \geq 0$, i que té exactament una casella bloquejada, ens demanen omplir la resta de la caselles amb les següents peces:

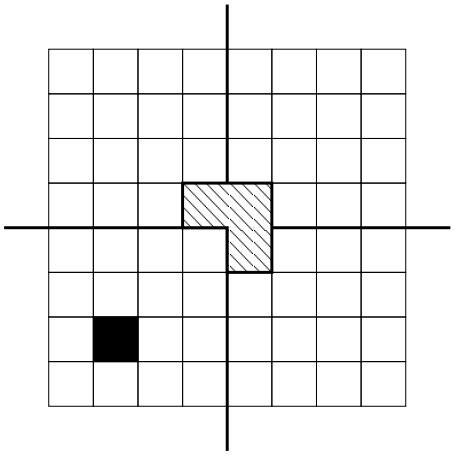


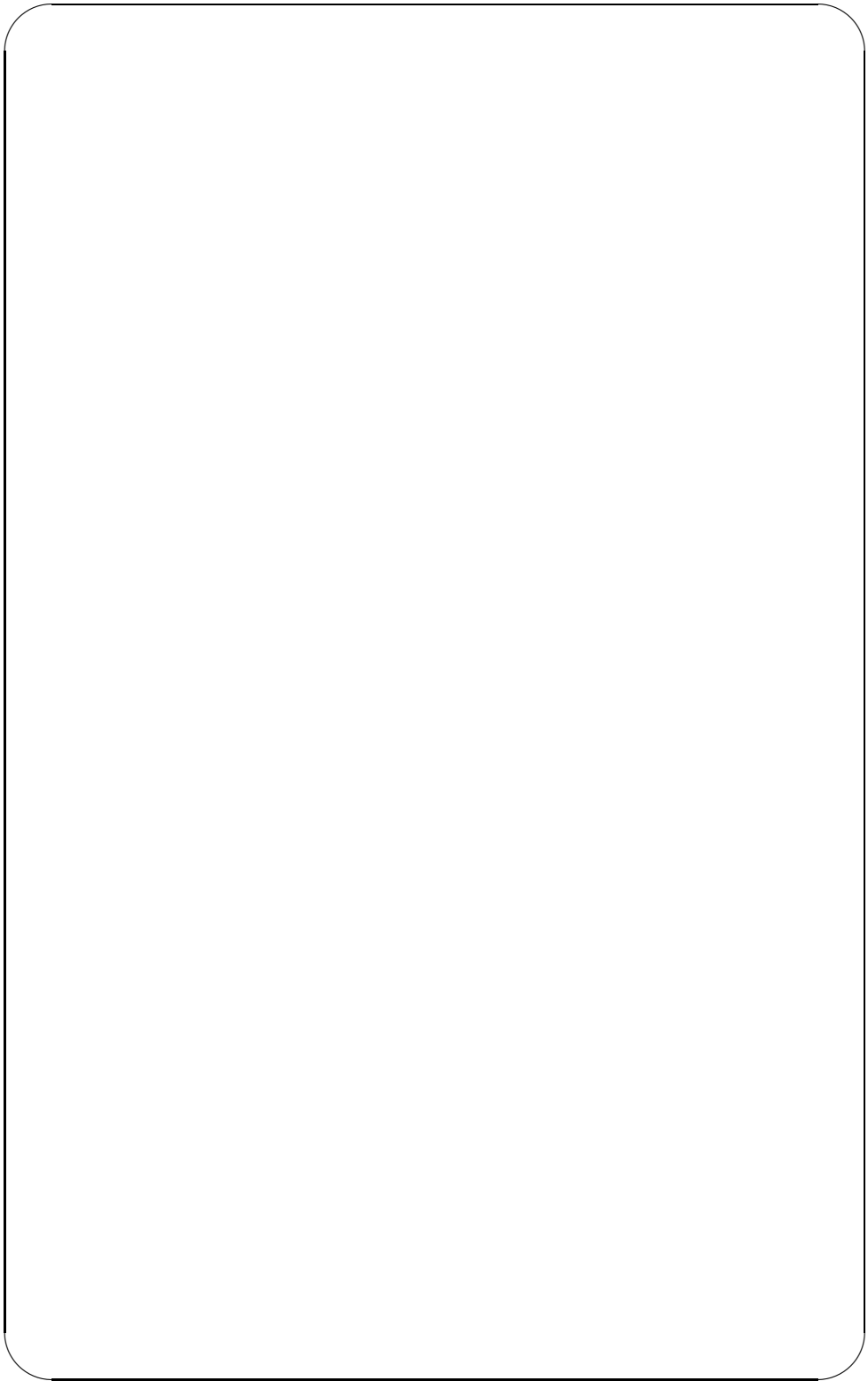
Com és d'esperar, les peces no es poden solapar ni sortir de la graella. Per exemple, per una graella 8×8 i la casella bloquejada en negre, una possible solució és:



(a) (2 pts.) Demostreu que per a tot $n \geq 0$, sigui quina sigui la posició de la casella bloquejada, sempre podrem omplir la resta de les caselles amb les peces indicades.

Pista: Observeu la següent figura.





Cognoms **Nom** **DNI**

(b) (2 pts.) Completeu el següent codi per tal de resoldre el problema plantejat. La matriu *M* representa la graella. Les files s'indexen de dalt a baix i les columnes d'esquerra a dreta.

```
void write_sol (const vector<vector<int>>& M);
typedef pair<int,int> Coord;

// Returns quadrant of pos in square [i_l,i_r] x [j_l,j_r]
// Quadrants are:
// 0 1
// 2 3
int quadrant(Coord pos, int i_l, int i_r, int j_l, int j_r) {
  int size = j_r - j_l + 1;
  int i_m = i_l + size/2;
  int j_m = j_l + size/2;
  if ( [                                    ] ) return 0;
  if ( [                                    ] ) return 1;
  if ( [                                    ] ) return 2;
  return 3;
}

void fill (vector<vector<int>>& M, int i_l, int i_r, int j_l, int j_r,
           Coord c_blocked, int& num){
  if (i_l == i_r) return; // 1x1
  int size = j_r - j_l + 1;
  int i_m = i_l + size/2; // Midpoints
  int j_m = j_l + size/2;

  vector<Coord> coords_blocked(4); // Blocked cell in each quadrant
  coords_blocked[0] = { [          ] , [          ] };
  coords_blocked[1] = { [          ] , [          ] };
  coords_blocked[2] = { [          ] , [          ] };
  coords_blocked[3] = { [          ] , [          ] };
  int q = quadrant(c_blocked, i_l, i_r, j_l, j_r);
  coords_blocked[q] = c_blocked;

  for (int k = 0; k < 4; ++k)
    if (M[coords_blocked[k].first][coords_blocked[k].second] == [      ])
      M[coords_blocked[k].first][coords_blocked[k].second] = [      ];
  ++num;

  fill(M, [                                    ], num); // Q0
  fill(M, [                                    ], num); // Q1
  fill(M, [                                    ], num); // Q2
  fill(M, [                                    ], num); // Q3
}
```

```
}

int main(){
  int n; cin >> n;
  int size = pow(2,n);
  Coord blocked;
  cin >> blocked.first >> blocked.second;
  vector<vector<int>> M(size,vector<int>(size,-1));

  M[blocked.first][blocked.second] = 0;
  // 0 initially occupied, -1 to be filled yet, n > 0 indicates piece identifier

  int num = 1; // All cells with the same num are part of the same piece
  fill(M, 0, size - 1, 0, size - 1, blocked, num);
  write_sol(M);
}
```

(c) (1 pt.) Quin és el cost del codi anterior en funció de n? I en funció del nombre de caselles?