1. Considere a função abaixo codificada.

```
int misterio (int v[], int n) {
  int cont=0 ;

  for (int i = 0; i < n; i++) {
     cont++ ;

     bool sing = true ;
     for (int j = i+1; j < n; j++)
        if (v[i] == v[j])
           sing = false ;

     if (!sing)
        cont-- ;
  }
  return cont ;
}
```

a) Diga o que faz o código acima.

b) Determine a complexidade temporal T(n) desta função. Justifique.

c) Proponha uma implementação mais eficiente.

2. Considere o seguinte método para processar uma sequência ordenada de números:

```
bool metodo (int A[], int valor, int tam) {
       int i = 0, j = tam-1 ;
       bool enc = false ;

       while (i < j && !enc) {
          if ((A[i] + A[j]) == valor)
             enc = true ;
          else {
             int mid = (i+j)/2 ;
             if (A[i]+A[j] > valor)
               j = mid-1  ;
             else
               i = mid+1 ;
          }
       }
       return enc ;
}
```

a) Explique o que faz o método acima apresentado.

b) Indique se o método é determinístico ou não e indique a sua complexidade temporal. Justifique.

3. Considere o seguinte programa que permite aplicar um filtro que "suaviza" uma imagem representada por uma matriz quadrada, na qual um valor inteiro representa um tom de cinzento.

```
#define N 1000

int filtro(int posi, int posj,int m[N][N])
{
   int soma=0;
   for (int i=posi-1;i<=posi+1;i++)
       for (int j=posj-1;j<=posj+1;j++)
          soma=soma+m[i][j];

   return soma/9;
}

void main() {

  int old_matrix[N][N];
  int new_matrix[N][N];

  preencher_matriz(old_matrix);   //Preencher valores matriz old_matrix

  //Aplicação do filtro
  for (int i=1;i<N-1;i++)
    for (int j=1;j<N-1;j++)
       new_matrix[i][j]=filtro(i,j,old_matrix);

}
```

Determine a complexidade temporal da aplicação do programa em questão. Justifique.