

CCM0128 – Computação II

CURSO DE CIÊNCIAS MOLECULARES

PRIMEIRO SEMESTRE DE 2025

Prova de Recuperação – 18/7/2025

Nome completo: _____

NUSP: _____

Assinatura: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. Preencha o cabeçalho acima.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade.
4. A prova consta de **3 questões**. Verifique antes de começar a prova se o seu caderno de questões está completo. **A prova vale 13 pontos**.
5. Não é permitido o uso de folhas avulsas para rascunho.
6. Não é permitido o uso de artefatos eletrônicos.
7. Não é permitida a consulta a livros, apontamentos ou colegas.
8. Não é necessário apagar rascunhos no caderno de questões.

DURAÇÃO DA PROVA: 2 horas

Questão	Nota
1	
2	
3	
Total	

Q1 (5.0 pontos) Considere o programa abaixo.

```
public class Q1
{
    public static Queue<Integer> mysteryQ(Queue<Integer> q) {
        StdOut.print("In: ");
        StdOut.println(q);

        if (q.size() <= 1)
            return q;

        Queue<Integer> q0 = new Queue<>();
        Queue<Integer> q1 = new Queue<>();

        int v = q.dequeue(); // pivot of partition
        while (!q.isEmpty()) {
            int x = q.dequeue();
            if (x <= v)
                q0.enqueue(x);
            else
                q1.enqueue(x);
        }

        q0 = mysteryQ(q0);
        q1 = mysteryQ(q1);

        for (int x : q0)
            q.enqueue(x);
        q.enqueue(v);
        for (int x : q1)
            q.enqueue(x);

        StdOut.print("Out: ");
        StdOut.println(q);

        return q;
    }

    public static void mystery(int[] a) {
        Queue<Integer> q = new Queue<>();
        for (int i = 0; i < a.length; i++)
            q.enqueue(a[i]);

        q = mysteryQ(q);

        for (int i = 0; i < a.length; i++)
            a[i] = q.dequeue();
    }
}
```

```

public static void main(String[] args)
{
    int[] a = StdIn.readAllInts();
    mystery(a);
    for (int i = 0; i < a.length; i++)
        StdOut.print(a[i] + " ");
    StdOut.println();
}
}

```

Diga qual será a saída do programa Q1 acima quando executado com os dígitos de seu NUSP. Por exemplo, se seu NUSP fosse 31415926, você teria de dizer a saída de Q1 quando executado com entrada

3 1 4 1 5 9 2 6

Importante: (a) Dê a saída de Q1 quando executado com os dígitos de **seu** NUSP. (b) Seu rascunho deve dar alguma indicação de como você obteve sua resposta.

Dica. Considere o seguinte programa:

```

public class Q1Example
{
    public static void main(String[] args)
    {
        // a[0] = 11, a[1] = 22, etc
        int[] a = {11, 22, 33, 44, 55};
        Queue<Integer> q = new Queue<>();

        for (int i = 0; i < a.length; i++)
            q.enqueue(a[i]);
        StdOut.println(q);

        for (int x : q)
            StdOut.print(x + " ");
        StdOut.println();
    }
}

```

Eis uma execução de Q1Example:

```

$ java-introcs Q1Example
11 22 33 44 55
11 22 33 44 55
$

```

Rascunho

Saída do programa

Q2 (4.0 pontos) Considere o seguinte programa.

```
public class Q2
{
    public static String divisores235(int N) {
        String t = "";
        if (N % 2 == 0) t += "1";
        else            t += "0";
        if (N % 3 == 0) t += "1";
        else            t += "0";
        if (N % 5 == 0) t += "1";
        else            t += "0";
        return t;
    }

    public static ST<String, SET<Integer>> similar235(int[] data, int a, int b) {
        ST<String, SET<Integer>> st = new ST<>();
        for (int i = a; i < b; i++) {
            String divisores235 = divisores235(data[i]);
            if (!st.contains(divisores235))
                st.put(divisores235, new SET<Integer>());
            st.get(divisores235).add(data[i]);
        }
        return st;
    }

    public static void main(String[] args)
    {
        // data[0] = 31, data[1] = 41, data[2] = 59, etc
        int[] data = {31, 41, 59, 26, 53, 58, 97, 93, 23, 84,
                     62, 64, 33, 83, 27, 95,  2, 88, 41, 97,
                     16, 93, 99, 37, 51,  5, 82,  9, 74, 94,
                     45, 92, 30, 78, 16, 40, 62, 86, 20, 89,
                     98, 62, 80, 34, 82, 53, 42, 11, 70, 68
                     };

        int i = Integer.parseInt(args[0]);
        int j = i + 20;

        ST<String, SET<Integer>> st = similar235(data, i, j);

        for (String divisores235 : st.keys()) {
            StdOut.print(divisores235 + ": ");
            for (int x : st.get(divisores235))
                StdOut.print(x + " ");
            StdOut.println();
        }
    }
}
```

Seja S a soma dos três últimos dígitos de seu NUSP.

(a) Seu NUSP: _____

(b) Valor de S : _____

(c) Diga qual será a saída do programa Q2 acima quando executado da seguinte forma:

```
$ java-introcs Q2 <valor de S>
```

Por exemplo, se seu NUSP fosse 31415926, teríamos $S = 17$, e assim você teria de dizer a saída da execução

```
$ java-introcs Q2 17
```

Importante: você deve dizer a saída de Q2 com seu valor de S .

Rascunho

Saída do programa

Q3 (4.0 pontos) Considere o seguinte programa.

```
public class Inversions {

    // merge and count
    private static long merge(int[] a, int[] aux, int lo, int mid, int hi) {
        long inversions = 0;

        for (int k = lo; k <= hi; k++)
            aux[k] = a[k];

        int i = lo, j = mid + 1;
        for (int k = lo; k <= hi; k++) {
            if (i > mid)
                a[k] = aux[j++];
            else if (j > hi)
                a[k] = aux[i++];
            else if (aux[j] < aux[i]) {
                a[k] = aux[j++];
                inversions += mid - i + 1;
            }
            else
                a[k] = aux[i++];
        }

        return inversions;
    }
}
```

```

// Returns the number of inversions in the subarray a[lo..hi].
// Side effect: a[lo..hi] is rearranged in ascending order.
private static long count(int[] a, int[] aux, int lo, int hi) {
    long inversions = 0;
    if (hi <= lo) return 0;
    int mid = lo + (hi - lo) / 2;
    inversions += count(a, aux, lo, mid);
    inversions += count(a, aux, mid + 1, hi);
    inversions += merge(a, aux, lo, mid, hi);
    return inversions;
}

/**
 * Returns the number of inversions in the integer array.
 * The argument array is modified.
 */
public static long count(int[] a) {
    int[] aux = new int[a.length];
    return count(a, aux, 0, a.length - 1);
}

/**
 * Reads a sequence of integers from standard input and
 * prints the number of inversions to standard output.
 */
public static void main(String[] args) {
    int[] a = StdIn.readAllInts();
    StdOut.println(count(a));
}
}

```

Seja a um vetor de inteiros. Lembre que uma inversão em a é um par de índices (i, j) tal que $0 \leq i < j < N$ e $a[i] > a[j]$, onde $N = a.length$.

- (a) Suponha que a função `merge()` em `Inversions` acima seja executada tendo como entrada vetores a e aux e $lo = 0$, $mid = \lfloor (N - 1)/2 \rfloor$, e $hi = N - 1$, onde $N = a.length = aux.length$. Naturalmente, supomos também que $a[0] \leq a[1] \leq \dots \leq a[mid]$ e que $a[mid + 1] \leq a[mid + 2] \leq \dots \leq a[N - 1]$. Prove que o valor devolvido por tal execução de `merge()` é o número de inversões (i, j) em a com $0 \leq i \leq mid$ e $mid < j < N$.

[illegible]

(b) Seja a o vetor construído em `main()` de `Inversions`. Prove que o valor impresso por `Inversions` é o número de inversões em a .

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

(c) Qual é a complexidade de tempo de **Inversions**? Esboce um argumento para justificar sua resposta.

(Rascunho)

(Rascunho)