CCM0118 – Computação I

CURSO DE CIÊNCIAS MOLECULARES

SEGUNDO SEMESTRE DE 2024

Prova Substitutiva – 12/12/2024

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 12 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 (4.0 pontos) Considere o seguinte programa.

```
public class Q1
{
    public static void mystery(int[] a) {
        StdOut.print(a[0]);
        mystery(a, 1);
        StdOut.println();
    }

    public static void mystery(int[] a, int i) {
        if (i == a.length)
            return;
        if (a[i - 1] >= a[i]) {
            StdOut.print(" " + a[i]);
            mystery(a, i + 1);
            return;
        }
        StdOut.print("/ " + a[i]);
        mystery(a, i + 1);
    }

    public static int[] digits(int N) {
        int[] t = new int[10];
        int i = 0;
        while (N > 0) {
            t[i] = N % 10;
            N /= 10;
            i++;
        }
        show(t);
        int[] d = new int[i];
        for (int j = i - 1; j >= 0; j--)
            d[i - 1 - j] = t[j];
        show(d);
        return d;
    }

    public static void show(int[] a) {
        int N = a.length;
        for (int i = 0; i < N; i++)
            StdOut.print(a[i] + " ");
        StdOut.println();
    }

    public static void main(String[] args)
    {
        int NUSP = Integer.parseInt(args[0]);
        int[] d = digits(NUSP);
        mystery(d);
    }
}
```

Diga qual será a saída do programa acima quando seu número USP é fornecido. Por exemplo, se seu NUSP fosse 12345678, você teria de dizer qual é a saída da execução

```
$ java-introcs Q1 12345678
```

Observação. Seu NUSP não é 12345678. Use seu NUSP.

Saída

Rascunho

Q2 (4.0 pontos) Uma matriz de Hadamard de ordem N é uma matriz N por N com entradas em $\{-1, 1\}$ tal que quaisquer duas de suas linhas coincidam em $N/2$ entradas e diferem em $N/2$ entradas. As seguintes três matrizes são matrizes de Hadamard (1×1 , 2×2 e 4×4):

1

1 1
1 -1

1 1 1 1
1 -1 1 -1
1 1 -1 -1
1 -1 -1 1

Por exemplo, note que a 1a. linha da terceira matriz é 1 1 1 1, enquanto que sua 2a. linha é 1 -1 1 -1. Essas duas linhas coincidem na 1a. e 3a. entradas, enquanto que elas diferem na 2a. e 4a. entradas.

Considere o seguinte programa incompleto:

```
public class Q2
{
    public static int[][] hadamard(int N) {

        // a completar

    }

    public static void show(int[][] M) {
        int N = M.length;
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++)
                StdOut.printf("%2d ", M[i][j]);
            StdOut.println();
        }
    }

    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        int[][] H = hadamard(N);
        show(H);
    }
}
```

O programa acima deve imprimir uma matriz de Hadamard de ordem N quando N é dado como argumento de linha de comando. Por exemplo,

Interpretamos esta sequência de 4 movimentos da seguinte forma: [0 1] significa que movemos o disco no topo da torre 0 para a torre 1; [0 2] significa que movemos o disco no topo da torre 0 para a torre 2 (e fazemos isso duas vezes); [1 2] significa que movemos o disco no topo da torre 1 para a torre 2. A sequência de configurações fica como segue:

```
0: 2 2 1
1:
2:
```

```
0: 2 2
1: 1
2:
```

```
0: 2
1: 1
2: 2
```

```
0:
1: 1
2: 2 2
```

```
0:
1:
2: 2 2 1
```

Considere o seguinte programa incompleto:

```
public class Q3
{
    public static void solve(int[] m) {
        int N = m.length;
        solve(m, N, 0, 2, 1);
        StdOut.println();
    }

    public static void solve(int[] m, int N, int A, int C, int B) {

        // a completar

    }

    public static void main(String[] args)
    {
        int[] m = StdIn.readAllInts();
        solve(m);
    }
}
```

Complete o programa acima de forma que possamos resolver o problema das torres de Hanói generalizado considerado nesta questão. Para resolver a instância com $m_1 = 1$ e $m_2 = 2$ acima, o usuário executaria Q3.java com a seguinte entrada:

1 2

Para resolver a instância em que $m_1 = 2$, $m_2 = 1$ e $m_3 = 3$, isto é,

0: 3 3 3 2 1 1
1:
2:

o usuário executaria Q3.java com a entrada

2 1 3

A saída de Q3.java poderia então ser

[0 2] [0 2] [0 1] [2 1] [2 1] [0 2] [0 2] [0 2] [1 0] [1 0] [1 2] [0 2] [0 2]

Sugestão. A função

```
public static void solve(int[] m, int N, int A, int C, int B)
```

deve receber as entradas $m_1, m_2, \dots$ no vetor m ($m[0] = m_1, m[1] = m_2, \dots$), e deve supor que a instância a ser resolvida tem N tipos diferentes de discos (m_1 com raio 1, $\dots$, m_N com raio N). Além disso, esta função deve supor que todos os $m_1 + \dots + m_N$ discos estão na torre A e queremos movê-los todos para a torre C , e B é a terceira torre. Finalmente, esta função deve imprimir a solução do problema especificado por esses parâmetros m, N, A, C e B .

Observação. A função `StdIn.readAllInts()` lê todos os inteiros na entrada padrão, e devolve um vetor com os inteiros lidos.

Escreva aqui a função `solve()` que falta por completo:

(Rascunho)