MAC0122 – Princípios de Desenvolvimento de Algoritmos

Engenharia de Computação

Segundo Semestre de 2023

Prova 3 – 13/12/2023

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 (5.0 pontos) Diga qual será a saída do programa abaixo quando executado com seu número USP como argumento de linha de comando, quando fornecemos o arquivo `tale.txt` (dado a seguir) na entrada padrão. Por exemplo, supondo que seu NUSP seja 31415926, você deve dizer qual é a saída da execução

```
java-introcs Q1 31415926 < tale.txt
```

Importante: seu rascunho deve indicar como você chegou a sua resposta.

```
public class Q1
{
    public static int[] digits(int NUSP) {
        Stack<Integer> s = new Stack<>();
        while (NUSP > 0) {
            int d = NUSP % 10;
            s.push(d);
            NUSP /= 10;
        }
        int[] t = new int[s.size()];
        for (int i = 0; i < t.length; i++)
            t[i] = s.pop();
        return t;
    }

    public static void main(String[] args)
    {
        int[] NUSP = digits(Integer.parseInt(args[0]));
        int N = NUSP.length;
        String[] tale = new String[10];

        for (int i = 0; i < 10; i++)
            tale[i] = StdIn.readLine();

        StdOut.println(tale[NUSP[2]]); // index 2
        String T = tale[NUSP[2]];
        for (int i = 3; i < 6; i++) {  // indices 3, 4, 5
            StdOut.println(tale[NUSP[i]]);
            T += " " + tale[NUSP[i]];
        }
        // words: array containing the words in T
        String[] words = T.split("\\s+");

        ST<String, Queue<Integer>> index = new ST<>();
        for (int i = 0; i < words.length; i++) {
            if (!index.contains(words[i]))
                index.put(words[i], new Queue<>());
            index.get(words[i]).enqueue(i);
        }
        for (String s : index)
            StdOut.println(s + " " + index.get(s));
    }
}
```

DICAS. O seguinte trecho de código

```
    String line = "it was the best";
    String[] words = line.split("\\s+");
    for (int i = 0; i < words.length; i++)
        StdOut.println(words[i]);
```

imprime

```
it
was
the
best
```

O seguinte trecho de código

```
    ST<String, Integer> tab = new ST<>();
    tab.put("it", 0);
    tab.put("was", 1);
    tab.put("the", 2);
    for (String s : tab)
        StdOut.println(s);
```

imprime

```
it
the
was
```

O seguinte trecho de código

```
    Queue<Integer> q = new Queue<>();
    q.enqueue(0);
    q.enqueue(1);
    q.enqueue(2);
    StdOut.println(q);
```

imprime

```
0 1 2
```

Este é o conteúdo do arquivo `tale.txt` (arquivo com exatamente 10 linhas):

```
it was the best of times
it was the worst of times
it was the age of wisdom
it was the age of foolishness
it was the epoch of belief
it was the epoch of incredulity
it was the season of light
it was the season of darkness
it was the spring of hope
it was the winter of despair
```

Rascunho

Saída do programa

Q2 (4.0 pontos) Considere as seguintes funções `mystery()`:

```
// We assume a[0] <= a[1] <= a[2] <= ...
public static int mystery(int x, int[] a) {
    int N = a.length;
    if (N == 0 || x > a[N - 1])
        return N;
    return mystery(x, a, 0, N - 1);
}

// We assume x <= a[hi] and lo <= hi
public static int mystery(int x, int[] a, int lo, int hi) {
    if (lo == hi)
        return hi;
    int mid = lo + (hi - lo) / 2;
    if (x <= a[mid])
        return mystery(x, a, lo, mid);
    else
        return mystery(x, a, mid + 1, hi);
}
```

Considere o vetor de inteiros

`a = {0, 3, 4, 4, 5, 6, 7, 7, 7, 9, 9, 10, 11, 12, 12, 14, 17, 17, 18, 20}`

com 20 inteiros em ordem não-decrescente.

(a) Qual é o valor devolvido pela chamada `mystery(50, a)`?

(b) Qual é o valor devolvido pela chamada `mystery(7, a)`?

(c) Qual é o valor devolvido pela chamada `mystery(13, a)`?

(d) Qual é o valor devolvido pela chamada `mystery(-50, a)`?

(e) O que é o valor devolvido por `mystery(x, a)`? Seja preciso em sua resposta.

(f) Justifique sua resposta do item anterior: por que `mystery(x, a)` devolve o valor especificado em sua resposta?

Q3 (3.0 pontos) Escreva um programa chamado `LastWords.java` que recebe um inteiro $k \geq 0$ como argumento de linha de comando e um texto na entrada padrão e que tem como saída as últimas k palavras do texto, separadas por espaço, todas em uma linha, na ordem em que elas foram lidas.

Considere, por exemplo, o arquivo `tale.txt` da Questão 1. Os exemplos abaixo ilustram o comportamento esperado de seu `LastWords.java`:

```
$ java-introcs LastWords 1 < tale.txt
despair
$ java-introcs LastWords 5 < tale.txt
was the winter of despair
$ java-introcs LastWords 10 < tale.txt
the spring of hope it was the winter of despair
$ java-introcs LastWords 15 < tale.txt
season of darkness it was the spring of hope it was the winter of despair
$ java-introcs LastWords 0 < tale.txt

$ java-introcs LastWords 200 < tale.txt
it was the best of times it was [ saída omitida ] winter of despair
$
```

Como ilustrado no último exemplo acima, caso o texto fornecido na entrada padrão tenha k palavras ou menos, a saída deve conter todo o texto.

IMPORTANTE

(a) Seu programa `LastWords.java` não deve impor nenhuma restrição artificial ou desnecessária no valor de k.

(b) Seu programa deve usar espaço não mais que proporcional a k. Aqui supomos que as palavras do texto de entrada têm comprimento no máximo alguma constante (por exemplo, todas as palavras têm comprimento no máximo 30).

(c) O tempo de execução de seu programa deve ser proporcional ao tamanho do texto de entrada.

(d) Leia cada palavra da entrada padrão com `StdIn.readString()`.

(e) Use, obrigatoriamente, um objeto do tipo `Queue<String>` em seu programa.

(Rascunho)

(Programa `LastWords.java`)

* * *

(Rascunho)