PROVA 2

MAC121 ALGORITMOS E ESTRUTURAS DE DADOS I

2o. ***SEMESTRE DE 2017***

Nome:

Número USP:

Instruções:

(*1*) Esta prova é individual.

(*2*) A prova consiste de **6** questões (contando a Questão 0 nesta página). Note que é possível tirar mais de 100 pontos sem fazer todas as questões `:-)`

(*3*) As respostas devem estar nos locais indicados.

(*4*) Não é permitido o uso de aparelhos eletrônicos de qualquer natureza.

(*5*) Não destaque as folhas deste caderno.

(*6*) Não use folhas avulsas para rascunho. Não é necessário apagar seus rascunhos.

(*7*) Não é permitido consultar nenhum material ou consultar colegas.

Assinatura:

Sua assinatura acima atesta a autenticidade e originalidade de seu trabalho e que você compromete-se a seguir o código de ética da USP em todas as suas atividades, incluindo esta prova.

Boa sorte!

Q	0	1	2	3	4	5	**Total**
Nota							

Q0. [5 pontos] Leia o conteúdo desta página e preencha os itens requisitados. Assine acima, e atente ao significado de sua assinatura.

Data: 2017/12/7, 9:09am

Q1. [25 pontos] Considere `Problem.java` dado abaixo.

```
public class Problem {
    private static class StringPair {
        private String s, t;
        private StringPair(String s, String t) { this.s = s; this.t = t; }
    }
    public static void main(String[] args) {
        Stack<StringPair> p = new Stack<StringPair>();
        p.push(new StringPair("", args[0]));
        while (!p.isEmpty()) {
            StringPair x = p.pop();
            int n = x.t.length();
            if (n == 0) { StdOut.println(x.s); continue; }
            for (int i = 0; i < n; i++)
                p.push(new StringPair(x.s + x.t.charAt(i),
                                      x.t.substring(0, i) + x.t.substring(i+1, n)));
        }
    }
}
```

(*i*) Dê a saída de `java-introcs Problem ab`

Resposta:

(*ii*) Dê a saída de `java-introcs Problem abc`

Resposta:

(*iii*) Diga quantas linhas são impressas ao executarmos `java-introcs Problem abcdabc`

Resposta:

Rascunho:

Q2. [20 pontos] Considere `Mystery.java` abaixo.

```
public class Mystery {
    public static void main(String[] args) {
        String[] words = StdIn.readAllStrings();
        ST<String, SET<Integer>> st = new ST<String, SET<Integer>>();
        for (int i = 0; i < words.length; i++) {
            String s = words[i];
            if (!st.contains(s)) st.put(s, new SET<Integer>());
            st.get(s).add(i);
        }
        int N = 0;
        for (String s : st.keys()) N += st.get(s).size();
        String[] t = new String[N];
        for (String s : st.keys())
            for (int i : st.get(s)) t[i] = s;
        for (int i = 0; i < N; i++) StdOut.print(t[i] + " ");
        StdOut.println();
    }
}
```

(i) Suponha que o arquivo `t.in` contém a linha

`it was the best of times`

Qual é a saída de `java-introcs Mystery < t.in`?

Resposta:

(ii) Seja `texto.in` um arquivo texto (por exemplo, `texto.in` poderia conter o texto de *Mody Dick* ou de *Os Lusíadas*). Descreva em poucas palavras o que é a saída de `java-introcs Mystery < texto.in`.

Resposta:

Rascunho:

Q3. [30 pontos]

(i) Uma árvore binária de busca A contém as chaves `1, 2, ..., 10`. Suponha que executamos a busca do elemento 5 em A e, nessa busca, examinamos a seguinte sequência de elementos: `1, 2, 10, 4, 8, 5`. Desenhe uma possível ABB A consistente com essa informação. A ABB A é única?

Resposta:

(ii) Para cada uma das sequências abaixo, diga se existe uma ABB consistente com elas (no sentido do item (i)):

(a) `4, 10, 8, 7, 5` *Resp.*:

(b) `1, 10, 2, 9, 3, 7, 4, 8, 6, 5` *Resp.*:

(c) `1, 2, 10, 4, 8, 5` *Resp.*:

(iii) Inserimos em uma ABB inicialmente vazia as chaves `A, D, E, J, M, Q, S, T` na seguinte ordem: `E, D, Q, A, J, M, T, S`. Desenhe a ABB resultante.

Resposta:

Rascunho:

(*iv*) O seguinte código aparece em `BST.java`:

```
public Iterable<Key> keys(Key lo, Key hi) {
    Queue<Key> queue = new Queue<Key>();
    keys(root, queue, lo, hi);
    return queue;
}
private void keys(Node x, Queue<Key> queue, Key lo, Key hi) {
    if (x == null) return;
    // StdOut.print(x.key + " ");
    int cmplo = lo.compareTo(x.key);
    int cmphi = hi.compareTo(x.key);
    if (cmplo < 0) keys(x.left, queue, lo, hi);
    if (cmplo <= 0 && cmphi >= 0) queue.enqueue(x.key);
    if (cmphi > 0) keys(x.right, queue, lo, hi);
}
```

Qual é a sequência de elementos que examinamos na ABB do item (*iii*) quando executamos `keys("F", "R")`? Isto é, se descomentássemos `StdOut.prin...`, o que seria impresso por aquele `print()`?

Resposta:

Rascunho:

Q4. [25 pontos]

(*i*) Desenhe todos os possíveis heaps diferentes com as chaves `A, A, B, C, D`. Faça o mesmo com as chaves `A A A B B`. Neste item, use heaps orientados a mínimos (isto é, na raiz deve estar a menor chave).

Resposta:

Resposta (continuação):

(*ii*) Considere o código abaixo.

```
public class Sort {
    public static void main(String[] args) {
        MinPQ<String> pq = new MinPQ<String>();
        String[] a = StdIn.readAllStrings();
        for (String s : a) pq.insert(s);
        while (!pq.isEmpty()) StdOut.println(pq.delMin());
    }
}
```

(*a*) Qual é o efeito de `Sort.java` em sua entrada?

Resposta:

(*b*) Por que é impossível implementar-se `MinPQ.java` usando-se apenas `compareTo()` de forma que ambos `insert()` e `delMin()` façam sempre $O(\log \log N)$ comparações?

Resposta:

Q5. [35 pontos] Considere `Quick3string.java` abaixo.

```
public class Quick3string {
    private static int count;
    public static void sort(String[] a) { sort(a, 0, a.length-1, 0); }
    private static int charAt(String s, int d) {
        if (d == s.length()) return -1;
        count++; // StdOut.println(s.charAt(d) + " in " + s);
        return s.charAt(d);
    }
    private static void sort(String[] a, int lo, int hi, int d) {
        if (hi <= lo) return;
        int lt = lo, gt = hi;
        int v = charAt(a[lo], d);
        int i = lo + 1;
        while (i <= gt) {
            int t = charAt(a[i], d);
            if      (t < v) exch(a, lt++, i++);
            else if (t > v) exch(a, i, gt--);
            else            i++;
        }
        StdOut.print("Index " + d + ": ");
        for (int j = lo; j <= hi; j++) StdOut.print(a[j] + " ");
        StdOut.println();
        sort(a, lo, lt-1, d);
        if (v >= 0) sort(a, lt, gt, d+1);
        sort(a, gt+1, hi, d);
    }
    private static void exch(String[] a, int i, int j)
    { String temp = a[i]; a[i] = a[j]; a[j] = temp; }
    public static void main(String[] args) {
        String[] a = StdIn.readAllStrings();
        sort(a);
        StdOut.print("Sorted: ");
        for (String s : a) StdOut.print(s + " ");
        StdOut.println("\nNumber of characters examined: " + count);
    }
}
```

(i) Ao executarmos `Quick3string.java` com a entrada

```
it was the best
```

obtemos a saída

```
Index 0: best it the was
Index 0: the was
Sorted: best it the was
Number of characters examined: 6
```

Suponha agora que descomentamos o comando `StdOut.prin...`. Este comando de impressão produzirá algumas linhas de saída. Diga quais são essas linhas explicitamente.

Resposta:

(*ii*) Executamos agora `Quick3string.java` com a entrada

```
now is the time for all good people
```

e obtemos a seguinte saída (a última linha da saída está omitida abaixo):

```
Index 0: is good all for now time people the
Index 0: good all for is
Index 0: all for good
Index 0: all for
Index 0: people time the
Index 1: the time
Sorted: all for good is now people the time
```

Diga qual é o valor final de `count` nessa execução. Como é a última linha da saída? (A linha que foi omitida acima.)

Resposta:

(*iii*) Suponha que fornecemos como entrada para `Quick3string.java` um arquivo texto contendo algo como 50.000 strings e 2.000.000 caracteres no total (comprimento médio dos strings: 40). É possível o valor final de `count` ser menor que 2.000.000?

Resposta:

Rascunho: