Universidade Federal do ABC
Bacharelado em Ciência da Computação
Algoritmos e Estrutura de Dados I - Prof. Fabrício Olivetti de França e Paulo Henrique Pisani
Turma A1/A2 Diurno - Prova B

NOME/RA : .....

Instruções: responda as seguintes questões na folha de prova. Entregue as duas folhas com nome e RA.

Questão 01 (1.0 pto). Um algoritmo de união de duas listas desordenadas, ambas de tamanho n, requer a operação de busca na segunda lista para cada elemento da primeira lista. Qual é a complexidade desse algoritmo em notação O?

R.: $O(n^2)$

Questão 02 (1.5 pto). Dada a seguinte estrutura de lista simplesmente ligada:

```
typedef struct linked_node linked_node;
struct linked_node {
    int data;
    linked_node *next;
};
```

Crie a função `void muda_lista(linked_node *inicio);` que remove todos os elementos de valor x ímpar e insere os elementos $x-1$ e $x+1$ no lugar e nessa ordem. Importante: a função **não** deve criar outra lista, apenas deve modificar a lista passada no parâmetro *inicio* e deve ser implementada na linguagem C.

```
void muda_lista(linked_node * inicio) {
    linked_node * newnode;
    if (inicio == NULL) return;

    if (inicio->data % 2 != 0) {
        inicio->data -= 1;
        newnode = malloc(sizeof(linked_node));
        newnode->data = inicio->data + 2;
        newnode->next = inicio->next;
        inicio->next = newnode;
        muda_lista(newnode->next);
    } else {
        muda_lista(inicio->next);
    }
}
```

Questão 03 (1.5 pto). Dada uma sequência de trens $1, 2, 3, 4$ e S representando a operação de empilhar e U de desempilhar, a sequência $SSUSSUUU$ transforma a ordem 1234 em 2431. Mostre a sequência de operações para transformar a ordem 123456 em 154632.

R.: `SUSSSSUUSUUU`

Questão 04 (1.0 pto). Um determinado sistema utilizará uma lista. Você precisa decidir entre implementá-la como uma array sequencial ou como uma lista ligada. Qual você escolheria? Quais fatores seriam levados em consideração?

R.: Para os casos em que a lista não cresce ou cresce em "blocos", e a operação mais frequente é a de busca, a array sequencial é mais interessante. Já nos casos em que inserção e remoção de elementos são as operações mais frequentes, a lista ligada é melhor.

Questão 05 (1.5 pto). O algoritmo de busca binária continuaria funcionando se trocássemos `left = mid + 1` por `left = mid`?

```
int * busca_bin (int k, int * x, int n) {
  int left = 0, right = n-1;
  int mid = (left+right)/2;

  while (x[mid] != k)
  {
    if (x[mid] > k) right = mid-1;
    else left = mid+1;

    mid = (left+right)/2;
    if (left > right) return NULL;
  }
  return &x[mid];
}
```

R.: Não funcionará, pois no caso em que `left = mid` e `mid = right - 1`, se `x[mid] < k`, `left` será atualizado para o mesmo valor e a condição de parada nunca será satisfeita.

Questão 06 (1.5 pto). Mostre que se um nó de uma árvore binária possui dois filhos, então seu sucessor não possui filho a esquerda e seu predecessor não possui filho a direita.

R.: considere a árvore

```
   8
  / \
 4   10
```

O sucessor atual de 8 é 10, se acrescentarmos um filho a esquerda de 10, só podemos inserir o 9, que passa a ser o sucessor de 8 e não possui filho a esquerda. Da mesma forma ao inserir um valor antecessor de 8 como filho de 4.

Questão 07 (1.0 pto). Suponha que temos os números de 1 a 1000 em uma árvore binária de busca e queremos buscar pelo número 521. Qual das seguintes sequências não podem ser a sequência de nós examinados:

a. $2, 252, 401, 531, 402, 450, 522, 521$
b. $900, 315, 321, 723, 621, 422, 500, 521$
c. $631, 123, 532, 224, 312, 351, 500, 521$
d. $2, 23, 151, 253, 612, 250, 261, 521$
e. $75, 99, 985, 123, 422, 523, 522, 521$

R.: (d)

Questão 08 (1.5 pto). Implemente um algoritmo na linguagem C que verifica se uma árvore binária é uma árvore binária de busca.

R.:

```
list * in_order(tree * t, linked_node * p) {

  if(t==NULL) return p;

  p = in_order(t->left, p);
  p = insere_fim(p, t->x);
  p = in_order(t->right, p);

  return p;
}

int eh_busca(tree * root) {
    /* se é vazia, é de busca */
    if(root==NULL) return 1;

    /* em uma arvore de busca, o percurso emordem deixa os elementos na ordem */
    linked_node *lista = NULL;
    lista = in_order(root);

    /* verifica se a lista está ordenada */
    while(lista->next != NULL) {
        if (lista->data > lista->next->data) return 0;
    }
    return 1;
}
```