PUC-Rio – Estruturas de Dados – INF1010
Prova 1 – Turma 3wb – 3/10/2016

Responda as questões abaixo nas folhas em separado distribuídas junto com a prova. As respostas podem ser escritas a lápis ou caneta. Indique claramente a questão que está respondendo e, quando cabível, os passos que seguiu para chegar a sua resposta.

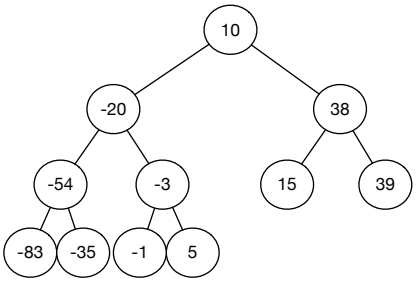
Por favor guarde seu celular/tablet/o_que_for antes de começar a prova e não o utilize até sair da sala.

1. (1,5 pontos) Considere a interface para operações sobre conjuntos de números inteiros vista em aula, esboçada abaixo:

```
typedef unsigned int Set;
/* cria um conjunto com n elementos */
Set* setCreate(void);
/* insere o elemento i no conjunto */
void setInsert(Set *set, int i);
/* remove o elemento i do conjunto */
void setRemove(Set *set, int i);
...
/* verifica de set2 e' um sub conjunto de set1 */
int setIsSubset( Set *set1,  Set *set2);
```

   Vamos supor uma implementação que contempla conjuntos de números naturais com valores entre 0 e 31, e que utiliza a representação por mapa de bits vista em sala. Implemente a função `setIsSubset` *sem utilizar* outras operações da interface apresentada.

2. (1 ponto) Suponha a árvore AVL mostrada a seguir.



   Como ficará a árvore se realizarmos cada uma das operações (cada uma das operações indicadas ocorre na árvore original mostrada). Desenhe (na folha a parte) a nova árvore para cada caso.

   (a) inserção com chave -100

   (b) inserção com chave -2

3. (1 ponto) Suponha que estamos implementando uma lista de prioridades usando um heap implementado com array, como visto em sala. Se soubermos que a ordem de inserção foi 3-10-21-8-12-20-30, como ficará o array? Preencha abaixo. E se a ordem for 10-20-8-3-30-21-12? Explique como chegou à sua resposta.

   com a inserção 3-10-21-8-12-20-30:

   com a inserção 10-20-8-3-30-21-12:

4. (2,5 pontos) Escreva uma função C que, dada uma árvore binária de busca m, com a representação interna abaixo e uma chave c, imprima todas as chaves *menores que c* presentes na árvore.

```
typedef struct smapa Mapa;
struct smapa {
  int chave;
  int dados;
  Mapa* esq;
  Mapa* dir;
};
...
int succ (Mapa *m, int c);
```

5. (1,5 pontos) Suponha uma implementação de AVL onde cada nó armazena o número de nós na subárvore que tem raiz nele. Escreva em C a operação `rotacao_a_direita` convencional e como fica o novo código da operação com esse novo campo em cada nó.

```
typedef struct smapa Mapa;
struct smapa {
  int chave;
  int conteudo;
  short int bf;
  int numnos;
  struct smapa* esq;
  struct smapa* dir;
};
...
static Mapa* rotacao_a_direita (Mapa *m);
```

6. (1,0 ponto) Dada a implementação de listas de prioridade vista em sala para um *max-heap*, responda, explicando suas respostas: (a) O custo de *consultar* quem é o maior elemento da lista varia com o número de elementos? Por que? Se sim, qual é o custo, no pior caso, em função de n, o número de elementos? (b) O custo de *remover* o maior elemento da lista varia com o número de elementos? Por que? Se sim, qual é o custo, no pior caso, em função de n, o número de elementos?

7. (1,5 pontos) Escreva uma função *não recursiva* para calcular a altura de uma árvore AVL:

```
int debug_altura (Mapa *m);
```

Boa prova!