Avaliação – Estruturas de Dados II

Leia atentamente todas as instruções do documento. Em caso de problemas nos enunciados das questões ou de necessidade de explicações adicionais, o documento será atualizado. Entre em contato em caso de dúvidas.

Regras Gerais

- Linguagens aceitas: C, C++, Haskell, Java e Python.
- Caso sejam necessárias instruções adicionais para executar o seu código, forneça um arquivo `README` ou utilize ferramentas de automação como `Make` ou `CMake` para facilitar os processos de compilação e execução.
- Códigos idênticos e/ou gerados por ferramentas de IA serão desconsiderados.
- **Comente todos os códigos com o máximo de detalhes possível**. Questões com códigos (não|mal) comentados receberão desconto de 1,0.

1. (2,5) [**Implementação de Árvore Red-Black**] Implemente uma árvore rubro-negra com dados numéricos informados e exiba as informações conforme exemplo abaixo:

Entrada

```
12 5 15 3 10 13 17 4 7 11 14 6 8
```

Saída

```
Percurso em ordem: 3 4 5 6 7 8 10 11 12 13 14 15 17
Raiz: 12
Altura: 4
Quantidade de nós vermelhos: 5
Quantidade de nós pretos: 8
```

Observações:

- Desconsidere nós nulos e a possibilidade da árvore informada ser nula.
- A correção deste exercício será automática, portanto, siga rigorosamente o padrão de entrada e saída para evitar erros de formatação. Não insira mensagens adicionais, faça somente o que o exercício pede.
- Eventuais espaços em branco após o último elemento da árvore na operação de percurso (primeira linha da saída) não precisam ser tratados.

2. (2,5) [**Implementação de Árvore Trie**] Implemente uma Trie para salvar as senhas presentes no arquivo `rockyou-menor.txt`[1]. O programa receberá consultas a prefixos e deve imprimir as senhas presentes no arquivo `rockyou-menor.txt` que coincidam com esses prefixos. Quando o padrão não for encontrado, o programa deve imprimir "`Nenhuma senha encontrada com esse prefixo.`". A entrada termina com EOF (*end of file*).

[1] gerado com `head -n 100000 rockyou.txt > rockyou-menor.txt` a partir da lista `rockyou.txt`

Entrada

```
teste
0123456789
!!!
```

Saída

```
tester
0123456789
01234567890
012345678910
!!!!!!
```

Observações:

- As consultas serão feitas somente com caracteres de código ASCII entre 32 e 126 (inclusive).
- Caso algum prefixo consultado coincida com o padrão de alguma senha que contenha caractere(s) fora do intervalo [32, 126], trate livremente esta situação (exemplos: salve a senha original ignorando o caractere, não salve a senha etc.).

3. (2,5) [**Implementação de Árvore B**] Implemente uma árvore B que comece vazia e realize as operações de inserção, impressão de percurso em ordem e busca, solicitadas na entrada pelo usuário. A entrada termina com EOF (*end of file*).

Entrada

```
insere 10
insere 20
insere 5
insere 6
insere 13
insere 30
insere 8
insere 18
percorre
busca 12
busca 13
```

Saída

```
Percurso da árvore B em ordem: 5 6 8 10 13 18 20 30
Valor 12 não encontrado!
Valor 13 encontrado!
```

Observações

- A árvore deve começar vazia.
- Assuma que não serão informadas entradas fora do padrão, ou seja, as entradas serão escritas corretamente e sempre com o número de parâmetros esperado.

- A ordem da árvore deve ser 2. Então o número mínimo de chaves é 1, o número máximo de filhos é 4 e o número mínimo de filhos é 2.
- Eventuais espaços em branco após o último elemento da árvore na operação de percurso não precisam ser tratados.

4. (2,5) [**Implementação de Árvores em Problema Interpretativo**] Escolha algum problema da plataforma Beecrowd e resolva-o utilizando árvore. Envie a URL do problema, o código da sua solução e uma captura de tela comprovando o aceite do seu exercício na plataforma.

Observação: Exceto o problema *Fake News*.