Como cada função foi testada

(I) Função tree()

Teste:

Inicializar arvore

Cria-se uma árvore, insere-se uma pergunta de início. Depois, verifica-se se os dados do current da árvore confere com as inseridas, econfere se o current é igual ao root (o current no início é sempre declarado igual ao root).

(II) Função insert ()

Recebe como parâmetros

- frase_entrada, uma string que representa a pergunta ou a resposta no nó da árvore.
- opcao_entrada, expressa se ele irá inserir no filho à direita(sim) ou à esquerda (não) .
- tipo_entrada, expressa o tipo de frase do nó.

Teste:

Operações de inserções no início

É testado se a inserção de uma pergunta inicial e suas respostas contruirá a árvore, seguindo a estrutura pergunta(pai) com filhos respostas. Desse modo, a saída será uma raiz com dois filhos, como verificado e confirmado pela saída de teste.

O jogo erra a resposta no início

É testado se o processo de inserção de uma pergunta no lugar de uma resposta errada(quando o usuário erra), e essa resposta e a resposta correta(inserida pelo usuário) são filhos dessa nova pergunta inserida no lugar certo(sim ou não, algo determinado pelo usuário) . Isso foi testado para todos os casos ocorrentes no início, ou seja, quando o programa erra quando o usuário escolhe sim, inserindo a resposta correta ou como sim ou não para a pergunta; ou quando o usuário escolhe não, seguindo a mesma lógica. Desse modo, tem-se 4 casos. E como verificado em todos os casos, não há erros.

O jogo erra no decorrer

Nesse caso de teste, o raciocínio é análogo. Porém, o erro ao invés de ocorrer no início, ocorrerá no decorrer, cuja ocorrência pode ocorrer para cada caso do erro de resposta no início, como descrito anteriormente. Assim, para cada dos 4 casos anteriores, teremos outros 4 casos, que é errar no sim ou nao, e para cada um a resposta entrar no sim ou não. Desse modo, teremos 16 casos de prosseguimento de jogo. O processo também prosseguiu sem problemas.

(III) Função eliminate()

Recebe como parâmetros
- Opcao_usuario

Deletar uma sub-árvore
Teste direcionado ao processo de deleção de sub-árvore. Nesse teste, cria-se uma árvore de profundidade 4, ela é printada no terminal no mecanismo pre-order transversal:

>>É um ser vivo?
>>martelo
>>É um mamífero?
>>cobra
>>Ele é quadrúpede?
>>humano
>>zebra

Depois, é chamado a função de deletar a sub-árvore à direita do filho à direita da raiz, eliminando portanto uma sub-árvore de profundidade 3 (no caso a pergunta da árvore do exemplo de teste é “Ele é quadripete?”). O resultado sai como o esperado, eliminando o node da pergunta e os nodes respostas vinculados a ele:

>>É um ser vivo?
>>martelo
>>É um mamífero?
>>cobra

(IV) Função save_game()

Recebe como parâmetros
- Nome do arquivo

Salvar a árvore

Uma mesma árvore ao teste anterior é formado. Ao se declarar uma string com o nome de arquivo teste “testar.txt”, a função save_game() da árvore é acionada. Desse modo, um arquivo de nome “testar.txt” é criado e salvo com as informações associadas à árvore. Para verificar o seu funcionamento, a função deve apresentar retorno 1 se ela funcionou. Além disso, verifica-se o arquivo gerado por esse processo, conferindo cada informação do jogo salvo às informações organizadas no arquivo(seguindo um padrão em que o primeiro campo corresponde ao índice do node; o segundo, o tipo de frase; o terceiro, a frase em si). Basta abrir o arquivo, para confirmar que os dados conferem.

(V) Função decodifica_nodes()

Recebe como parâmetros

- ind_entrada, string correspondente ao indice do node
- tipo_entrada, string correspodente ao tipo de frase do node
- frase_entrada, string correspondente à frase em si

Decodificar o arquivo em árvore

A função testa a validez do mecanismo usado pela função da decodificação do arquivo em árvore, garantindo que cada atributo relacionado à árvore corresponda ao que está descrito no arquivo. No caso, a frase contida na forma decodificada é extraída e diferenciada como um atributo isolado(ex. “resposta” e” humano” extraído).

(VI) Função ler_arquivo()

Recebe como parâmetros

- O nome_arquivo, o nome do arquivo que será carregado

Ler o arquivo

O teste consiste em abrir o arquivo “testar.txt” anteriormente criado e a partir dela criar a árvore por meio da função ler_arquivo(). Depois, a árvore criada é salva no arquivo “Outro_jogo.txt”. A partir desses 2 arquivos, verifica-se a equivalência entre elas, já que ambas representam a mesma árvore. Desse modo, cada arquivo é aberto e cada linha é analisada entre ambas. Se houver divergência, algo inesperado e incorreto, o programa reporta erro. Porém, como requerido e esperado, a função não reportou erro, provando a validez tanto da função ler_arquivo quanto save_game. Para uma maior veracidade, recomenda-se abrir cada arquivo txt mencionado e compará-las, e é verificado a equivalência.