Especificações algébricas de tipos de dados abstractos

1. Construa uma especificação algébrica para o tipo *int* (inteiros) com as operações *zero*, *suc* (sucessor), *pred* (predecessor), *soma*, *subtracção* e *multiplicação*.

2. Construa uma especificação algébrica para o tipo *bool* (booleanos) com as operações *true*, *false*, *not*, *and*, *or*, *if*, *ifthenelse*, *equiv*, e *xor* (disjunção exclusiva).

3. Construa uma especificação algébrica para o tipo *pilha* de *elem* com as operações *nova* (constante, que representa uma pilha sem elementos), *sobrepõe* (binária, que permite colocar um novo elemento na pilha) , *topo* (unária, que permite calcular o último elemento colocado na pilha), *retira* (unária, que permite retirar o último elemento colocado na pilha), *vaziaQ* (unária, que verifica se uma pilha tem ou não elementos), *alterna* (binária, que dadas duas pilhas constrói uma nova pilha que alterna os elementos das duas), *sobrepõegen* (binária, que dadas duas pilhas sobrepõe a primeira à segunda). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

4. Construa uma especificação algébrica para o tipo *fila-de-espera* de *elem* com as operações *inicial* (constante, que representa uma fila sem elementos), *entra* (binária, que acrescenta um elemento no fim da fila), *primeiro* (unária, que permite calcular o primeiro elemento da fila), *sai* (binária, que permite retirar o primeiro elemento da fila), *vaziaQ* (unária, que verifica se uma fila tem ou não elementos), *desiste* (unária, que dada uma fila retorna a mesma se esta tiver no máximo um elemento e caso contrário a fila que se obtém quando o último elemento sai) e *junta* (binária, que dadas duas filas retorna a fila que se obtém quando se coloca a segunda fila no fim da primeira). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

5. Construa uma especificação algébrica para o tipo *arv-binária* de *elem* com as operações *nova* (constante, que representa uma árvore vazia), *constr* (ternária que, dado um elemento e e duas árvores r e s, constrói a árvore com e no nó raiz, r como subárvore esquerda e s como subárvore direita), *raiz* (unária, que permite calcular o elemento do nó raiz), *arvesq* (unária, que permite calcular a subárvore esquerda), *arvdir* (unária, que permite calcular a subárvore direita) e *vaziaQ* (unária, que verifica se a árvore é ou não vazia). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

6. Construa uma outra especificação algébrica para o tipo *arv-binária* de *elem* com as operações, *leaf* (unária, que transforma um elemento numa árvore singular, ou seja, com um único nó), *left* (binária, que dado um elemento e e uma árvore s, constrói a árvore que tem e como raiz e s como sua subárvore esquerda), *right* (binária, que dado um elemento e e uma

árvore s, constrói a árvore que tem e como raiz e s como sua subárvore direita), *both* (ternária, que dado um elemento e e duas árvore r e s, constrói a árvore que tem e como raiz, s como sua subárvore esquerda e r como sua subárvore direita), *breadth* (unária, que permite calcular o número de folhas), *edge* (unária, que permite calcular o número de arestas), *nodes* (unária, que permite calcular o número de nós), *singQ* (unária, que verifica se a árvore é ou não singular), *degnQ* (unária, que verifica se a árvore tem ou não uma única folha), e *depth* (unária, que permite calcular a profundidade da árvore). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias, em particular, a operação *max* (máximo entra naturais).

7. Construa uma especificação algébrica para o tipo *sequência* de *elem* com as operações *vazia* (constante, que representa uma sequência vazia), *acrf* (binária, que permite acrescentar um elemento no fim da sequência), *acri* (binária, que permite acrescentar um elemento no início da sequência), *retiraf* (binária, que permite retirar um elemento do fim da sequência), *retirai* (binária, que que permite retirar um elemento do início da sequência), *último* (unária, que permite calcular o último elemento da lista), *primeiro* (unária, que permite calcular o primeiro elemento da lista), *concat* (binária, que constrói a concatenação de duas sequências) e *inverte* (unária, que permite inverter uma sequência). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

8. Construa uma outra especificação algébrica para o tipo *sequência* de *elem* com as operações *empty* (constante, que representa uma sequência vazia), *make* (unária, que transforma um elemento numa sequência), *ladd* (binária, que permite acrescentar um elemento no início da sequência), *radd* (binária, que permite acrescentar um elemento no fim da sequência), *concat* (binária, que constrói a concatenação de duas sequências), *lenght* (unária, que calcula o comprimento de uma sequência), *emptyQ* (unária, que verifica se a sequência é ou não vazia) e *seq-const-of-lenght* (binária, que dado um elemento e e um natural n constrói a sequência de comprimento n que só tem e's). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

9. Construa uma especificação algébrica para o tipo *lista* de *elem* com as operações *nil* (constante, que representa uma lista vazia), *add* (binária, que dado um elemento e uma lista acrescenta esse elemento no fim da lista), *head* (unária, que calcula o primeiro elemento da lista), *tail* (unária, que retira o primeiro elemento da lista), *delete* (binária, que dado um elemento e e uma lista que retira a última ocorrência de e da lista), *remove* (binária, que dadas duas listas remove da segunda lista a última ocorrência de cada elemento que está presente na primeira lista), *concat* (binária, que constrói a concatenação de duas listas), *isin* (binária, que dado um elemento e uma lista verifica se o elemento está na lista), *partof*

(binária, que dadas duas listas verifica se cada elemento da primeira lista tem pelo menos o mesmo número de ocorrências em ambas as listas), *permutation* (binária, que dadas duas listas verifica se as duas listas são permutações uma da outra). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

10. Especifique o tipo de dados *mapa* que inclua para além de uma constante *vazio* (identificando o mapa vazio), as seguintes operações: *reg* ($reg(x, y, m)$ regista no mapa m em x o valor y), *anul* ($anul(x, m)$ anula o registo no mapa m de todos os valores associados com x); *val* ($val(x, m)$ retorna o último valor associado a x no mapa m).

11. Especifique o tipo de dados *matriz* 2×2 (de inteiros) que inclua as operações *matnul* (constante, que contrói a matriz nula), *unit* (constante, que contrói a matriz diagonal unitária), *matrix* (quaternária, que dados 4 inteiros constrói a matriz cuja primeira linha são os dois primeiros argumentos e cuja segunda linha são os dois últimos argumentos), *add* (adição), *sub* (subtracção) e *mult* (multiplicação).

12. Construa uma especificação algébrica para o tipo *saco* de naturais com as operações *newbag* (constante, que representa o saco vazio) *insbag* (binária, que dado um elemento e um saco, permite inserir uma cópia do elemento no saco) *delbag* (binária, que dado um elemento e um saco, permite apagar uma cópia do elemento no saco), *delallbag* (binária, que dado um elemento e um saco, permite apagar todas as cópias do elemento no saco), *emptybagQ* (unária, que verifica se o saco está ou não vazio), *inbagQ* (binária, que dado um elemento e um saco, verifica se existe ou não uma cópia do elemento no saco), *nbag* (binária, que dado um elemento e um saco, permite calcular o número de cópias do elemento que existem no saco), *tbag* (unária que permite calcular o número de elementos que existem no saco) e *dbag* (unária que permite calcular o número de elementos distintos que existem no saco). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.

13. Especifique o tipo de dados *vector ilimitado* de booleanos com as operações *empty* (constante, que representa um vector vazio), *assign* (ternária, que dado um vector, um booleano e um natural, permite inserir o booleano na posição indicada pelo natural) e *read* (binária, que dado um vector e um natural, permite calcular o booleano no posição indicada pelo natural). Nota: introduza também a especificação de tipos (auxiliares) que considere relevantes, com as operações que considere necessárias.