Conjuntos com Tabelas Hash

Neste trabalho você irá implementar um tipo abstrato de dados que lida com conjuntos de elementos, utilizando tabelas hash. Os elementos são valores inteiros entre -2^16 e 2^16. Um conjunto não deve conter elementos duplicados. Podem haver no máximo 50 conjuntos diferentes em um dado momento. Seu programa irá receber operações da entrada padrão e retornará resultados na saída padrão. Cada linha de entrada especifica uma operação. As operações obedecerão a seguinte sintaxe e especificação:

1) `criar()`
   Cria um conjunto vazio. Retorna um número inteiro de identificação do conjunto criado ou -1 em caso de erro.
   A identificação deve ser sequencial, iniciando em 0 (primeiro conjunto).

2) `inserir(elemento, conjunto)`
   Insere o elemento no conjunto especificado. Retorna 1 em caso de sucesso e -1 em caso de erro.
   Tentar inserir elemento duplicado não deve causar erro.
   Tentar inserir em conjunto que não existe deve retornar erro.

3) `existe(elemento, conjunto)`
   Verifica se o elemento pertence ao conjunto especificado. Retorna 1 se existe e -1 se não existe.
   Tentar verificar a existência de um conjunto que não existe retorna -1.

4) `excluir(elemento, conjunto)`
   Remove o elemento do conjunto especificado. Retorna 1 em caso de sucesso e -1 em caso de erro.
   Tentar remover um elemento que não existe causa erro.
   Tentar remover de um conjunto que não existe NÃO causa erro.

5) `unir(conjunto1, conjunto2)`
   Faz a união dos dois conjuntos especificados, retornando o identificador de um novo conjunto contendo todos elementos de ambos conjuntos (sem duplicações).
   Este novo conjunto não deve reaproveitar identificadores criados previamente.
   Os conjuntos de entrada continuam existindo.

6) `listar(conjunto)`
   Lista todos elementos do conjunto especificado, em ordem crescente, separados por vírgula em uma única linha.

7) `fim()`
   Encerra o programa.

Sua implementação deve utilizar tabelas hash como principal estrutura subjacente, onde cada conjunto é representado por uma tabela hash. O tamanho inicial de cada tabela deve ser de 50 posições, obrigatoriamente. Você deve decidir por todos os demais detalhes de implementação.

Avaliação

A avaliação do trabalho considerará a correção das respostas, a qualidade do código, a quantidade de memória utilizada e o tempo de execução de um conjunto de operações.

Correção das respostas
No processo de avaliação seu programa receberá um conjunto de operações de forma automatizada, com diferentes níveis de dificuldade. Portanto, é essencial que a especificação tanto das entradas como das saídas seja seguida a risca.

Qualidade do código
Serão avaliadas boas práticas de construção de código, incluindo nome de variáveis e funções, identação, presença de comentários úteis etc.

Quantidade de memória
Será avaliado o uso de memória máximo pelo programa na execução das operações. Menos memória é melhor. Uso excessivo de memória será penalizado (um desvio padrão abaixo da média).

Tempo de execução
Será medido o tempo de execução de um conjunto grande de operações. Tempos excessivamente longos (dois desvios padrões abaixo da média) serão penalizados. Os melhores tempos (um desvio padrão acima da média) receberão bônus na nota, exceto se fizerem uso excessivo de memória. O melhor tempo (que não faz uso excessivo de memória) receberá premiação especial.

Observação: apenas será medido tempo e espaço de programas que tiverem *todas* respostas corretas.

Outras Especificações e Submissão

O trabalho deverá ser implementado em C ou C++. Não deve depender de bibliotecas além das bibliotecas padrões. Não deve utilizar a biblioteca STL. Toda entrada deverá ser lida da entrada padrão e toda saída ser escrita na saída padrão.

Será submetido pelo AVA um único arquivo ZIP, denominado NomeSobrenome.zip, contendo todos códigos, *makefile* para geração do executável e um arquivo README.txt contendo

quaisquer bugs conhecidos do programa (bugs não listados terão desconto maior). O código deve ser compilável em plataforma Linux Ubuntu padrão. O executável gerado deverá se chamar "hash".

Exemplo de entrada	**Saída para o exemplo de entrada**
criar() inserir(10,0) inserir(20,0) excluir(10,0) existe(10,0) criar() inserir(30,1) unir(0,1) listar(2) fim()	0 1 1 1 -1 1 1 2 20,30

Código de Honra

O envio do trabalho para correção pressupõe que o código submetido é original e desenvolvido completamente pelo aluno que está submetendo. Pressupõe também que uma quantidade suficiente de esforço foi colocada no trabalho, não contendo erros banais ou catastróficos e que todas funcionalidades foram implementadas.

Desvios das pressuposições acima ocasionarão penalidades que podem variar desde a não avaliação do trabalho até a subtração de pontos da média final em casos mais drásticos (cópia de trabalho, por exemplo). É aceitável que o trabalho contenha erros não previstos, não é aceitável que não execute operações simples.

Trabalhos submetidos com nome NomeSobrenome.zip de forma literal serão desconsiderados :P