COMP LabBook ERE 2021 1

Lucas M. Schnorr

March 25, 2022

Contents

1 E3

1.1 Comentários por grupo

1.1.1 **DONE** GrupoA

- Submetido em atraso.
- Bom gerenciamento de memória.
- Estranhamente, temos um erro de sintática reportado no teste w22 embora a entrada esteja correta. Mais estranhamente, o teste w27 (que contém um `while` também) funciona sem problemas! O que pode estar acontecendo com o teste w22?
- Para algumas entradas de teste, o programa gera uma AST diferente daquela esperada. Vejamos:

```
[1] "w10" "w11" "w12" "w13" "w37" "w39" "w63" "w65" "w67" "w72" "w78"
```

- Inversão do operador de shift na AST (w10, . . .)!
- Problemas de precedência de operadores (w37, . . .)
- Limpeza da fecha aspas demasiada agressiva no final (w63)
- Sobre a Sec 2.1
  - Consta na E3 que o valor léxico deve ser atribuído, além de literais e identificadores (realizados), também para caracteres especiais e operadores compostos.

- Sobre a Sec 2.2 / E3
  - Não entendi porque existem três campos para guardar os filhos do nós desta AST n-ária: `filhos, pai,proximo_com`

CODE PL.

- Sobre o arquivo token.[ch]
  - Cadê a identificação dos autores? Quem implementou estes arquivos?
- Sobre o arquivo `parser.y`
  - Quem implementou as ações semânticas?

1.1.2 DONE GrupoB

- Por alguma razão, o programa não imprime na saída padrão o label de alguns nós da AST. Tendo em vista que a avaliação automática é realizada tomando-se por base que aquela saída (vide Sec 2.4) é um reflexo da estrutura de dados da AST, os testes falham. Isso pode ser observado rapidamente nos testes inicias: w01, w02, w09, mas se repentem para testes subsequentes.
- O nó output da AST precisa ter pelo menos um filho que é o seu parâmetro. Isso foi objeto de uma dúvida divulgado no fórum (pergunta também para o input) como w08.
- A inicialização de variáveis no momento da declaração deve fazer parte da AST (w10, w12, por exemplo)
- A indexação de vetor deve estar na AST também (w11)
- Alguns casos são mais difíceis de identificador o problema (w19)
- Os testes demonstram que as regras de precedência também não foram implementadas corretamente conforme o teste ilustrativo w78 mostra que a multiplicação tem precedência sobre a soma, ainda que em alguns casos esteja correto (w73) mas daí é por que a associatividade à esquerda "simula" a precedência correta.
- Os testes que tem problemas na árvore AST gerada são estes:

```
 [1] "w01" "w02" "w03" "w04" "w08" "w09" "w10" "w11" "w12" "w13" "w18" "w19"
[13] "w20" "w21" "w23" "w24" "w25" "w26" "w27" "w30" "w63" "w64" "w65" "w66"
[25] "w67" "w68" "w70" "w71" "w75"
```

- Sobre a Sec 2.1
  - Consta na E3 que o valor léxico deve ser atribuído para literais, identificadores, caracteres especiais e operadores compostos. Mas na solução enviada, temos também para algumas palavras reservadas (aquelas que finalmente acabam indo para a AST).
  - Veja que nas regras que usam esses tokens, o valor léxico é desconsiderado. Por exemplo, nas regras dos NTs input, output.
- Sobre a Sec 2.2
  - Consta que a estrutura de dados deve prever uma quantidade de filhos arbitrário (uma árvore n-ária). Mas na implementação, `MAX_FOLHAS` tem o valor 4.
- Os testes abaixo falham por falta de aderência ao descrito na Sec 2.6 da E3. Sugiro executar o `valgrind` para resolver esses problemas conforme o descrito no Anexo B da E3. Sugiro empregar o parâmetro `--leak-check=full`.

```
 [1] "w01" "w02" "w03" "w04" "w05" "w06" "w07" "w08" "w09" "w10" "w11" "w12"
[13] "w13" "w14" "w15" "w16" "w17" "w18" "w19" "w20" "w21" "w22" "w23" "w24"
[25] "w25" "w26" "w27" "w28" "w29" "w30" "w31" "w32" "w33" "w34" "w35" "w36"
[37] "w37" "w38" "w39" "w40" "w41" "w42" "w43" "w44" "w45" "w46" "w47" "w48"
[49] "w49" "w50" "w51" "w52" "w53" "w54" "w55" "w56" "w57" "w58" "w59" "w60"
[61] "w61" "w62" "w63" "w64" "w65" "w66" "w67" "w68" "w69" "w70" "w71" "w72"
[73] "w73" "w74" "w75" "w76" "w77" "w78"
```

- Conforme detalhado na E3, os testes objetivos passam se a AST está correta e se o valgrind reporta nenhum vazamento de memória.
- Interessante o comentário "//MUDAR TUDO DAQUI PRA BAIXO"
  - O que significa?

CODE PL.

- Sobre o arquivo ast.h e parte inicial do ast.c
  - Quem implementou estas regiões de códigos?

1.1.3 **DONE** GrupoC

- Erro sintático em w68? Por quê?
- Veja a Sec 2.1 / E3: lá indica que não há necessidade de implicar valor léxico para todos os tokens. Veja que mesmo declarados (parser.y), somente um subconjunto (de acordo com a E3) é de fato definido.

1.1.4 **DONE** GrupoD

- Submetido em atraso
- O programa não compila por conta desse problema:

```
/usr/bin/ld: main.o:/home/schnorr/ensino/private/comp/e3/GrupoD/main.c:10:
multiple definition of `arvore';
parser.tab.o:/home/schnorr/ensino/private/comp/e3/GrupoD/main.h:4: first defined here
```

  - Que indica que o nome arvore foi definido múltiplas vezes. Veja que jamais devemos definir variáveis globais nos arquivos de cabeçalho (.h), como foi feito na linha 4 de main.h.
  - Para compilar, o professor executou antes de `make`:

```
sed -i -e '4d' main.h # remove a quarta linha de main.h
```

- Uma visão geral dos resultados dos testes automáticos indica que os testes falham duplamente: por fato da AST gerada estar diferente mas também pelo fato do teste com valgrind reportar problemas de liberação de memória.
- Olhando de maneira subjetiva unicamente para as diferenças da AST (usando como base a estrutura da árvore impressa na saída padrão do programa), temos as seguintes observações
  - a lista de funções não foi corretamente implementada, gerando como no teste w01 vários grafos separados. Isso pode ser observado em outros testes que tem mais de uma função como no w03.
  - Chamada de função não foi implementada na AST (não aparece na saída)
  - A ligação da lista de comandos à função à qual ela pertence também não foi implementada (w04).
  - O comando simples de atribuição não há manifestação na AST (w05), embora por vezes funcione parcialmente (w06). Não entendi muito bem a diferença entre os dois.
  - Outros comandos simples também não foram implementados.
- Conforme vimos na especificação E3, não faz sentido lidar com `declaracao_var_local` do ponto de vista da AST, pois declarações não vão para a AST (somente as inicializações).
- Alguns problemas de programação (alocação de memória):
  - Por exemplo, na função `adiciona_nodo` temos `malloc(extra+sizeof(nodo)*8)`, não há necessidade do `extra` pois depois é feito o strdup (que na prática é um `malloc` como inicialização de dados), e a multiplicação por 8 também não se entende pois implica que se está alocando 8 estruturas do tipo `nodo`. Por quê??
- Quase a integralidade dos testes falham com o valgrind. Lançando o valgrind da seguinte forma (exemplo com o teste w03):

```
valgrind --leak-check=full ./etapa3 < w03
```

  - As seguintes alocações são jamais liberadas (lista não exaustiva):
    * `atribui_yylval` (scanner.l:96)
    * yylex (scanner.l:71)
    * yylex (scanner.l:86)
    * `adiciona_nodo` (ast.c:14)
- Sobre o exporta / estrutura de dados em árvore
  - Não vejo necessidade de ter duas funções de exportar (uma para nós, outro para as arestas). Poderia ser uma única função que exportaria, ao visitar um nó, o seu label e suas arestas. Não há necessidade que os nós e seus labels apareçam primeiro na saída.
  - Ao explorar essa vertente (o professor mesclou as duas funções colocando o printf do label dentro do `imprime_aresta` ficou claro que os endereços de memória (ou seja os filhos) registrados no vetor de ponteiros `filhos` não é o mesmo dos nós da recursão. Isso acontece porque deveria haver somente uma estrutura para nós, ou seja `LseNodo` não deveria existir. As arestas usam o endereço de memória de uma estrutura, mas os labels são impressos com endereços de memória de outra.
  - Em nossa AST, um nó não precisa saber seu irmão, basta saber quem são seus filhos.

1.1.5 DONE GrupoE

- Sobre a Sec 2.2
  - Consta que a estrutura de dados deve prever uma quantidade de filhos arbitrário (uma árvore n-ária). Mas na implementação, `MAX_FOLHAS` tem o valor 4.

1.1.6 DONE GrupoG

- O programa não compila, com erros de compilação em inúmeros arquivos parser.y, scanner.l e ast.c. Como isso passou desapercebido? Como o programa não compila, a submissão recebe nota zero.

1.1.7 DONE GrupoH

- Sobre a Sec 2.1
  - Consta na E3 que o valor léxico deve ser atribuído para literais, identificadores, caracteres especiais e operadores compostos. Mas na solução enviada, temos também para algumas palavras reservadas (aquelas que finalmente acabam indo para a AST).
  - Veja que nas regras que usam esses tokens, o valor léxico é simplesmente convertido para nó da AST. Nestes casos, seria melhor simplesmente criar o nó da AST diretamente (simplificando o léxico, o programa de uma maneira geral).

1.1.8 DONE GrupoI

- Colocar nome dos membros do grupo em todos os arquivos (incluindo .c)
- Alguns testes permanecem com problemas de memória:

```
[1] "w03" "w09" "w14" "w68" "w74"
```

Problemas de liberação:

- O malloc da linha token.c:8
- O malloc da linha tree.c:8
- O programa ainda detecta erro sintático, mas todas as entradas estão corretas. Estas são as entradas nesse ponto:

```
[1] "w30" "w41"
```

- Relacionados com o operador ternário e operador unário '?'.
- AST com erros:

```
[1] "w14" "w30" "w41"
```

- Return precisa do seu parâmetro na árvore (w14)
- Sobre a Sec 2.1
  - Consta na E3 que o valor léxico deve ser atribuído para literais, identificadores, caracteres especiais e operadores compostos. Mas na solução enviada, temos também para algumas palavras reservadas (aquelas que finalmente acabam indo para a AST).
  - Veja que nas regras que usam esses tokens, o valor léxico é desconsiderado. Por exemplo, nas regras dos NTs comandoIf.
- Sobre a Sec 2.2
  - Consta que a estrutura de dados deve prever uma quantidade de filhos arbitrário (uma árvore n-ária). Mas na implementação, `MAXIMO_FILHOS` tem o valor 4.

CODE PL.

- Sobre o arquivo tree.h e parte inicial do tree.c
  - Quem implementou estas regiões de códigos?
  - Cadê a identificação dos autores nestes arquivos?

1.1.9 DONE GrupoJ

- Por alguma razão há uma inversão no operador de -shift=. A AST não reflete o shift operador utilizado no código de entrada (w10, w11, w12, w13).
- Alguns vazamento de memória permanecem. Eles aparecem nos seguintes testes:

```
 [1] "w10" "w11" "w12" "w13" "w50" "w51" "w52" "w53" "w56" "w57" "w75" "w77"
[13] "w78"
```

- Executando com valgrind –leak-check=full, temos (o strdup da linha scanner.l:23 é o culpado - ou seja - o que faltou liberar):

```
==131680== 3 bytes in 1 blocks are definitely lost in loss record 1 of 1
==131680==    at 0x483F7B5: malloc (vg_replace_malloc.c:381)
==131680==    by 0x491631A: strdup (strdup.c:42)
==131680==    by 0x10A2F7: composite_operators (scanner.l:23)
==131680==    by 0x10A80B: yylex (scanner.l:84)
==131680==    by 0x10CA0F: yyparse (parser.tab.c:1259)
==131680==    by 0x10E31F: main (main.c:16)
```

- A AST para alguns testes não é a esperada (veja a lista de testes abaixo)
- Os argumentos de uma função devem fazer parte da AST (w03, w09, w68)
-
```
[1] "w03" "w09" "w10" "w11" "w12" "w13" "w68" "w74"
```

- Sobre a análise do código
- No arquivo bison, não empregou-se `%union` em favor de `api.value.type` para ter somente um valor semântico para todos os elementos (tokens, não terminais). Não vejo problema nesse ponto específico. No entanto, ao observar a estrutura em si (types.h) observa-se que o tipo de dado do nó da AST é um campo de `valor_lexico`, o que não faz muito sentido pois a AST não é um token, é algo independente da léxica (ainda que nas diretivas %type se faça referência direta ao campo `node` daquela union. Minha recomendação é então seguir o que está descrito na Sec 2.1 / E3.
- Por que não usar algo parecido como a função `composite_operators()` para os demais tokens que possuem valor léxico, evitando repetição de código. A função poderia ser parametrizável.
- Muito legal a implementação da árvore n-ária, com uma função n-ária! ;-) `create_node`. Legal!

1.1.10 DONE GrupoK

- Bom gerenciamento de memória!
- No entanto, subexistem testes que tem problemas na árvore AST gerada:

```
[1] "w01" "w03" "w09" "w29" "w64" "w68" "w69" "w75" "w76" "w77" "w78"
```

- Para a w01 (lista de funções), vários nós extras aparecem na árvore, e algumas vezes temos conexões com '(nil)' indicando um problema potencialmente grave.
- Idem (conexões com nil) para todas as outras entradas salvo w64
- O teste w64 falha pois as aspas simples não foram removidas ;-(

1.1.11 DONE GrupoM

- Os arquivos dentro do etapa3.tgz devem estar na raiz
  - Revisem as regras gerais por gentileza
- Não houve preocupação com o descrito na Sec 2.6 pois todos os testes falham devido a falta de liberação correta de memória.
- As árvores estão erradas para as entradas w39, w37, w65, w67, w78, porque as regras de precedência em expressões aritméticas estão incorretas
- Compilando os fontes com -g além do -c e executando o valgrind com `--leak-check=full` conforme descrito na E3 torna possível perceber que memória alocada na linha 50 do arquivo ast.c, na linha 111 do scanner.l, na linha 98 do ast.c, na lista 107 de ast.c nunca é liberada.

1.1.12 DONE GrupoN

- A solução não segue a especificação pois na saída (especificada na Sec 2.4), consta que o nome do nó deve ser especificado com o atributo `label` e não `txt` como foi submetido.
- O programa ainda detecta erro sintático, mas todas as entradas estão corretas. **Imagino que as importantes alterações necessárias após a correção da E2 não foram levadas em conta**. Estas são as entradas nesse ponto:

```
 [1] "w17" "w18" "w19" "w20" "w21" "w22" "w23" "w24" "w25" "w26" "w27" "w28"
[13] "w29" "w54" "w55" "w68" "w69" "w75" "w77" "w78"
```

- Os demais testes falhos (veja a listagem abaixo), são referentes a problemas de precedência de operadores (resquício da E2, mas agora verificados automaticamente).

```
 [1] "w01" "w02" "w03" "w04" "w05" "w06" "w07" "w08" "w09" "w10" "w11" "w12"
[13] "w13" "w14" "w15" "w16" "w30" "w31" "w32" "w33" "w34" "w35" "w36" "w37"
[25] "w38" "w39" "w40" "w41" "w42" "w43" "w44" "w45" "w46" "w47" "w48" "w49"
[37] "w50" "w51" "w52" "w53" "w56" "w57" "w58" "w59" "w60" "w61" "w62" "w64"
[49] "w65" "w66" "w67" "w70" "w71" "w72" "w73" "w74" "w76"
```

- Enfim o teste w63 acontece por conta de uma "Double free" na função libera. Sugiro executar o programa com o gdb para localizar exatamente a linha. Okay, eu fiz isso para vocês, o double free acontece na linha tree.c:83. Uma execução com o `valgrind` indicará onde aconteceu o primeiro free. Para essa entrada, o valgrind também reporta leituras de espaços de memória (4 bytes) jamais inicializados (o que pode levar a um problema).
- Sobre a Sec 2.1
  - Consta na E3 que o valor léxico deve ser atribuído, além de literais e identificadores (realizados), também para caracteres especiais e operadores compostos (ou seja, também para as ERs da linha 64 até a 97 do scanner.l).
- Uma dica, na criação de um nó da AST (por exemplo, para literal true), ao invés de fazer um `sprintf` da string estática "true" e passar isso para `create_node`, podes usar o próprio valor léxico (era o intuito da Sec 2.1 / E3). Isso evitaria também chamadas do tipo

  ```
  create_node("<=",NULL)
  ```

  pois ali o valor léxico do operador composto já terá a string.
- Sobre a Sec 2.2 / E3
  - Não entendi porque existem três campos para guardar os filhos do nós desta AST n-ária: `children`, `parent`, `next_cmd`.

CODE PL

- Sobre os arquivos `token_info.[ch]`
  - Cadê a identificação dos autores? Quem implementou estes arquivos?
- Sobre o arquivo `parser.y`
  - Quem implementou as ações semânticas?
- Sobre os demais arquivos
  - Cadê a identificação dos autores?

1.1.13 DONE GrupoO

- Submetido em atraso
- O programa não compila por conta desse problema:

```
/usr/bin/ld: /tmp/ccBu38Pr.o:(.bss+0x0): multiple definition of `Table'; /tmp/ccI22UiT.o:(.
/usr/bin/ld: /tmp/ccQxR6lF.o:(.bss+0x0): multiple definition of `Table'; /tmp/ccI22UiT.o:(.
/usr/bin/ld: /tmp/ccg6anxm.o:(.bss+0x40): multiple definition of `Table'; /tmp/ccI22UiT.o:(
/usr/bin/ld: /tmp/ccnr84Z6.o:(.bss+0x0): multiple definition of `Table'; /tmp/ccI22UiT.o:(.
/usr/bin/ld: /tmp/ccnr84Z6.o:(.bss+0x13878): multiple definition of `arvore'; /tmp/ccI22UiT
```

  - Que indica que os nomes Table e arvore foram definidos múltiplas vezes. Veja que jamais devemos definir variáveis globais nos arquivos de cabeçalho (.h), como foi feito na linha no arquivo has.h (Table). Estes são os comandos para corrigir o problema

```
sed -i 's#Hash_Node \*Table#extern Hash_Node *Table#' hash.h
sed -i "10i Hash_Node \*Table[HASH_SIZE];" hash.c
sed -i "s#AST \*arvore;#extern AST *arvore;#" parser.y
```

- Os operadores compostos e especiais não receberam valores léxicos
- A árvore não é n-ária (veja Sec 2.2, últimas frases)
- A especificação descrita na Sec 2.4 / E3 não foi implementada.
- A função main foi modificada (há uma `hashPrint` presente)
  - Não havia necessidade de usar uma hash em nossa E3
  - Reler a especificação E3
- Como a objetiva se baseia na saída do que a Sec 2.4 especifica, todos os testes automáticos falham.
- Observando as ações semânticas de `criaNodo` no parser.y, teço as seguintes observações
  - Muitas construções são desnecessárias (novamente, sugiro reler a E3), como declarações de variáveis globais, locais (salvo aquelas de inicialização que estas sim devem ser mantidas)
  -
- Quase a integralidade dos testes falham com o valgrind. Lançando o valgrind da seguinte forma (exemplo com o teste w03):

  ```
  valgrind --leak-check=full ./etapa3 < w03
  ```

  - As seguintes alocações são jamais liberadas (lista não exaustiva):
    * criaNodo (AST.c:15)
    * criaNodo (AST.c:23)
  - O programa tem várias escritas de dados em regiões de memória não mais alocadas (foi feito um free, e mesmo assim escrito). Sugiro no momento de fazer o free, imediatamente após, zerar o ponteiro atribuindo para este o valor NULL. Ao fazer isso, o programa dará segfault, mas permitirá corrigir os problemas.

1.1.14 DONE GrupoP

- Implementação em C++ (para registro)
- Bom gerenciamento de memória!
- No entanto, subexistem testes que tem problemas na árvore AST gerada:

```
[1] "w06" "w11" "w13" "w33" "w34" "w71" "w73" "w74"
```

- w06/w11/w13/w33/w34/w71, o identificador ficou faltando como filho do []
-

1.1.15 DONE GrupoT

- Sobre a Sec 2.1 / E3
  - O campo do `yylval` não se chama `valor_lexico` conforme a Sec 2.1.
  - Veja que não é recomendável alterar o conteúdo apontado pelo ponteiro `yytext` conforme podemos encontrar na linha do `scanner.l` onde se lê `yytext[size-2]`. Não entendi como esse código ali é capaz de remover as aspas duplas do literal string.
  - Não se trata de `COMPLEX_OPERATOR` mas de operador composto.
  - Palavras reservadas não devem ter valor léxico de acordo com a Sec 2.1 da E3, mas na função `CreateMiddleNode` está se criando um `LexicalValue` para palavra reservada. Veja que o "label" associado ao nó da AST pode ser simplesmente uma string tal qual recebida como parâmetro por esta função.
- Sobre a Sec 2.2 / E3
  - Não entendi porque existem três campos para guardar os filhos do nós desta AST n-ária: `Children`, `Next`, `NextCommand`.
  - Sobre os tipos desses nós da AST, percebe-se o uso de `FUNCTION_LIST` e outras listas. Mas de acordo com a especificação Sec 2.3 / E3, devemos ter listas mas cada nó da lista é o elemento listado. Ou seja, não deve haver explicitamente um nó do tipo lista. Para funções, por exemplo, é a própria função que aponta para a próxima (juntas elas formaram uma lista, mas cada nó da lista é uma função).

- Estranho algumas funções que lidam com nós da AST receberem ponteiro para ponteiro. Por quê isso é necessário? A minha recomendação é que isso é desnecessário (conhecendo o que se espera para a E3) e evita teres que fazer `&$1` por exemplo.

- Sobre a Sec 2.3 / E3
  - Veja os estes automáticos falhos.
  - Uma análise rápida das ações no `parser.y` está okay (embora não possa ser avaliada com o suporte da parte objetiva).
  - Sugestão de tratar todos os filhos de cada nó de maneira isonômica, ou seja, sem diferenciação entre os tipos de filhos (se é Children, se é Next, se é NextCommand.
  - Sugestão de evitar alocar/desalocar coisas que sabe-se não serão utilizadas, tais como uma boa parte de símbolos simples.

- Sobre a Sec 2.4 / E3
  - Embora o código esteja presente, não foi possível - sem modificar o programa - averiguar o funcionamento das funções de exportar a árvore tendo em vista que o programa sempre lança o problema de `double free` para a quase totalidade das entradas de teste utilizadas na avaliação objetiva, salvo o teste `w00` que é o programa vazio. Para unicamente averiguar a aderência à Sec 2.4 da E3, o professor comentou todas as chamadas a função `free` no programa (somente no arquivo `ast.c`) e lançou o programa novamente. Desta forma, pode-se observar falha de segmentação na maioria dos testes, sendo considerada correta apenas a saída da primeira entrada, que é o programa vazio.

1.2 Objetivo

1.2.1 Sumário

```
read_csv("e3/e3_output.csv", col_types=cols())
```

Grupo	Objetivo
GrupoE	10
GrupoH	10
GrupoC	9.87
GrupoI	8.99
GrupoP	8.99
GrupoK	8.61
GrupoA	8.48
GrupoJ	7.85
GrupoB	0.13
GrupoD	0.13
GrupoN	0.13
GrupoT	0.13
GrupoG	0
GrupoM	0
GrupoO	0

1.2.2 Análise dos diversos erros encontrados

1. Erros sintáticos ou léxicos

   Os seguintes grupos tem erros sintáticos ainda que todas as entradas fornecidas estão lexicalmente e sintaticamente válidas de acordo com as especificações E1 e E2.

Grupo	test	Output
GrupoA	w22	line 4: syntax error, unexpected $\text{TOKEN}_{\text{ERRO}}$
GrupoC	w68	line 3 - syntax error, unexpected $\text{TK}_{\text{LITINT}}$
GrupoI	w30	[3]: syntax error, unexpected $\text{TOKEN}_{\text{ERRO}}$
GrupoI	w41	[4]: syntax error, unexpected $\text{TOKEN}_{\text{ERRO}}$
GrupoN	w17	error in line 6: syntax error
GrupoN	w18	error in line 7: syntax error
GrupoN	w19	error in line 8: syntax error
GrupoN	w20	error in line 9: syntax error
GrupoN	w21	error in line 6: syntax error
GrupoN	w22	error in line 5: syntax error
GrupoN	w23	error in line 8: syntax error
GrupoN	w24	error in line 8: syntax error
GrupoN	w25	error in line 9: syntax error

Continued on next page

Continued from previous page

Grupo	test	Output
GrupoN	w26	error in line 6: syntax error
GrupoN	w27	error in line 5: syntax error
GrupoN	w28	error in line 3: syntax error
GrupoN	w29	error in line 3: syntax error
GrupoN	w54	error in line 5: syntax error
GrupoN	w55	error in line 5: syntax error
GrupoN	w68	error in line 3: syntax error
GrupoN	w69	error in line 3: syntax error
GrupoN	w75	error in line 3: syntax error
GrupoN	w77	error in line 3: syntax error
GrupoN	w78	error in line 3: syntax error

2. Erros de falha de segmentação

   Os programas dos seguintes grupos terminam por falha de segmentação com as entradas especificadas.

Grupo	test	Output
GrupoB	w55	Segmentation fault

3. Erros de liberação dupla de regiões de memória (double free)

   Os programas dos seguintes grupos terminam pela detecção da chamada de `free` a uma região já previamente liberada com a mesma função.

Grupo	test	Output
GrupoN	w63	Double free detected
GrupoT	w01	Double free detected
GrupoT	w02	Double free detected
GrupoT	w03	Double free detected
GrupoT	w04	Double free detected
GrupoT	w05	Double free detected
GrupoT	w06	Double free detected
GrupoT	w07	Double free detected
GrupoT	w08	Double free detected
GrupoT	w09	Double free detected
GrupoT	w10	Double free detected
GrupoT	w11	Double free detected
GrupoT	w12	Double free detected
GrupoT	w13	Double free detected
GrupoT	w14	Double free detected
GrupoT	w15	Double free detected
GrupoT	w16	Double free detected
GrupoT	w17	Double free detected
GrupoT	w18	Double free detected
GrupoT	w19	Double free detected
GrupoT	w20	Double free detected
GrupoT	w21	Double free detected
GrupoT	w22	Double free detected
GrupoT	w23	Double free detected
GrupoT	w24	Double free detected
GrupoT	w25	Double free detected
GrupoT	w26	Double free detected
GrupoT	w27	Double free detected
GrupoT	w28	Double free detected
GrupoT	w29	Double free detected
GrupoT	w30	Double free detected
GrupoT	w31	Double free detected
GrupoT	w32	Double free detected
GrupoT	w33	Double free detected
GrupoT	w34	Double free detected
GrupoT	w35	Double free detected
GrupoT	w36	Double free detected
GrupoT	w37	Double free detected
GrupoT	w38	Double free detected

Continued on next page

Continued from previous page

Grupo	test	Output
GrupoT	w39	Double free detected
GrupoT	w40	Double free detected
GrupoT	w41	Double free detected
GrupoT	w42	Double free detected
GrupoT	w43	Double free detected
GrupoT	w44	Double free detected
GrupoT	w45	Double free detected
GrupoT	w46	Double free detected
GrupoT	w47	Double free detected
GrupoT	w48	Double free detected
GrupoT	w49	Double free detected
GrupoT	w50	Double free detected
GrupoT	w51	Double free detected
GrupoT	w52	Double free detected
GrupoT	w53	Double free detected
GrupoT	w54	Double free detected
GrupoT	w55	Double free detected
GrupoT	w56	Double free detected
GrupoT	w57	Double free detected
GrupoT	w58	Double free detected
GrupoT	w59	Double free detected
GrupoT	w60	Double free detected
GrupoT	w61	Double free detected
GrupoT	w62	Double free detected
GrupoT	w63	Double free detected
GrupoT	w64	Double free detected
GrupoT	w65	Double free detected
GrupoT	w66	Double free detected
GrupoT	w67	Double free detected
GrupoT	w68	Double free detected
GrupoT	w69	Double free detected
GrupoT	w70	Double free detected
GrupoT	w71	Double free detected
GrupoT	w72	Double free detected
GrupoT	w73	Double free detected
GrupoT	w74	Double free detected
GrupoT	w75	Double free detected
GrupoT	w76	Double free detected
GrupoT	w77	Double free detected
GrupoT	w78	Double free detected

4. Erros de vazamento de memória

   De acordo com a Sec 2.6 da E4, "Gerenciar corretamente a memória". O programa valgrind foi utilizado para averiguar a correta liberação de toda a memória no final da execução. Na tabela abaixo, estão todos os testes que falharam em tal verificação. A coluna VG consiste no somatório da quantidade de bytes não liberada no final, e a coluna VGTEXT mostra a saída numérica do valgrind. Um exemplo representativo do comando utilizado na avaliação assim como sua saída:

```
$ valgrind ./etapa3 < ~/w19 $test 2>&1 | grep -v 0x | grep -A5 LEAK\ SUMMARY
==170160== LEAK SUMMARY:
==170160==    definitely lost: 56 bytes in 1 blocks
==170160==    indirectly lost: 536 bytes in 4 blocks
==170160==      possibly lost: 0 bytes in 0 blocks
==170160==    still reachable: 212 bytes in 1 blocks
==170160==         suppressed: 0 bytes in 0 blocks
```

   Para o exemplo acima, a coluna VGTEXT terá `56 536 0 212 0` e a coluna VG terá o valor `804`, portanto o teste falhou no aspecto vazamento de memória. Segue a tabela completa:

Grupo	test	VG	VGTEXT
GrupoB	w01	120	102 18 0 0 0
GrupoB	w02	146	134 12 0 0 0
GrupoB	w03	254	230 24 0 0 0
GrupoB	w04	280	60 186 0 34 0

Continued on next page

Continued from previous page

Grupo	test	VG	VGTEXT
GrupoB	w05	18	18 0 0 0 0
GrupoB	w06	179	85 94 0 0 0
GrupoB	w07	24	24 0 0 0 0
GrupoB	w08	114	78 0 0 36 0
GrupoB	w09	380	344 36 0 0 0
GrupoB	w10	330	104 226 0 0 0
GrupoB	w11	307	169 138 0 0 0
GrupoB	w12	330	104 226 0 0 0
GrupoB	w13	307	169 138 0 0 0
GrupoB	w14	21	21 0 0 0 0
GrupoB	w15	16	16 0 0 0 0
GrupoB	w16	22	22 0 0 0 0
GrupoB	w17	34	34 0 0 0 0
GrupoB	w18	78	74 4 0 0 0
GrupoB	w19	54	54 0 0 0 0
GrupoB	w20	98	94 4 0 0 0
GrupoB	w21	92	88 4 0 0 0
GrupoB	w22	34	34 0 0 0 0
GrupoB	w23	166	152 14 0 0 0
GrupoB	w24	82	78 4 0 0 0
GrupoB	w25	142	132 10 0 0 0
GrupoB	w26	176	160 16 0 0 0
GrupoB	w27	118	102 16 0 0 0
GrupoB	w28	18	18 0 0 0 0
GrupoB	w29	8	8 0 0 0 0
GrupoB	w30	30	30 0 0 0 0
GrupoB	w31	28	28 0 0 0 0
GrupoB	w32	36	36 0 0 0 0
GrupoB	w33	185	91 94 0 0 0
GrupoB	w34	185	91 94 0 0 0
GrupoB	w35	48	48 0 0 0 0
GrupoB	w36	48	48 0 0 0 0
GrupoB	w37	48	48 0 0 0 0
GrupoB	w38	48	48 0 0 0 0
GrupoB	w39	72	72 0 0 0 0
GrupoB	w40	28	28 0 0 0 0
GrupoB	w41	28	28 0 0 0 0
GrupoB	w42	28	28 0 0 0 0
GrupoB	w43	28	28 0 0 0 0
GrupoB	w44	28	28 0 0 0 0
GrupoB	w45	28	28 0 0 0 0
GrupoB	w46	36	36 0 0 0 0
GrupoB	w47	36	36 0 0 0 0
GrupoB	w48	36	36 0 0 0 0
GrupoB	w49	36	36 0 0 0 0
GrupoB	w50	76	70 6 0 0 0
GrupoB	w51	76	70 6 0 0 0
GrupoB	w52	76	70 6 0 0 0
GrupoB	w53	76	70 6 0 0 0
GrupoB	w54	36	36 0 0 0 0
GrupoB	w56	76	70 6 0 0 0
GrupoB	w57	76	70 6 0 0 0
GrupoB	w58	34	34 0 0 0 0
GrupoB	w59	34	34 0 0 0 0
GrupoB	w60	34	34 0 0 0 0
GrupoB	w61	48	48 0 0 0 0
GrupoB	w62	48	48 0 0 0 0
GrupoB	w63	297	70 186 0 41 0
GrupoB	w64	280	60 186 0 34 0
GrupoB	w65	276	264 12 0 0 0
GrupoB	w66	272	264 8 0 0 0
GrupoB	w67	276	264 12 0 0 0
GrupoB	w68	260	236 24 0 0 0
GrupoB	w69	4	4 0 0 0 0
GrupoB	w70	559	116 407 0 36 0

Continued on next page

Continued from previous page

Grupo	test	VG	VGTEXT
GrupoB	w71	235	137 98 0 0 0
GrupoB	w72	50	50 0 0 0 0
GrupoB	w73	48	48 0 0 0 0
GrupoB	w74	253	155 98 0 0 0
GrupoB	w75	157	139 18 0 0 0
GrupoB	w76	334	236 98 0 0 0
GrupoB	w77	69	63 6 0 0 0
GrupoB	w78	77	71 6 0 0 0
GrupoD	w01	76	72 0 0 4 0
GrupoD	w02	1563	119 1440 0 4 0
GrupoD	w03	697	160 533 0 4 0
GrupoD	w04	27	23 0 0 4 0
GrupoD	w05	362	94 264 0 4 0
GrupoD	w06	55	51 0 0 4 0
GrupoD	w07	232	90 138 0 4 0
GrupoD	w08	233	90 139 0 4 0
GrupoD	w09	1897	184 1709 0 4 0
GrupoD	w10	36	32 0 0 4 0
GrupoD	w11	54	50 0 0 4 0
GrupoD	w12	36	32 0 0 4 0
GrupoD	w13	54	50 0 0 4 0
GrupoD	w14	22	18 0 0 4 0
GrupoD	w15	22	18 0 0 4 0
GrupoD	w16	22	18 0 0 4 0
GrupoD	w17	655	122 529 0 4 0
GrupoD	w18	1057	130 923 0 4 0
GrupoD	w19	1063	138 921 0 4 0
GrupoD	w20	1465	146 1315 0 4 0
GrupoD	w21	1846	138 1704 0 4 0
GrupoD	w22	648	114 530 0 4 0
GrupoD	w23	1865	154 1707 0 4 0
GrupoD	w24	1065	138 923 0 4 0
GrupoD	w25	1871	162 1705 0 4 0
GrupoD	w26	2646	154 2488 0 4 0
GrupoD	w27	1448	130 1314 0 4 0
GrupoD	w28	374	106 264 0 4 0
GrupoD	w29	38	34 0 0 4 0
GrupoD	w30	763	102 657 0 4 0
GrupoD	w31	506	106 396 0 4 0
GrupoD	w32	646	114 528 0 4 0
GrupoD	w33	652	121 527 0 4 0
GrupoD	w34	652	121 527 0 4 0
GrupoD	w35	920	126 790 0 4 0
GrupoD	w36	920	126 790 0 4 0
GrupoD	w37	920	126 790 0 4 0
GrupoD	w38	920	126 790 0 4 0
GrupoD	w39	1468	150 1314 0 4 0
GrupoD	w40	506	106 396 0 4 0
GrupoD	w41	506	106 396 0 4 0
GrupoD	w42	506	106 396 0 4 0
GrupoD	w43	506	106 396 0 4 0
GrupoD	w44	506	106 396 0 4 0
GrupoD	w45	506	106 396 0 4 0
GrupoD	w46	646	114 528 0 4 0
GrupoD	w47	646	114 528 0 4 0
GrupoD	w48	646	114 528 0 4 0
GrupoD	w49	646	114 528 0 4 0
GrupoD	w50	646	110 532 0 4 0
GrupoD	w51	646	110 532 0 4 0
GrupoD	w52	646	110 532 0 4 0
GrupoD	w53	646	110 532 0 4 0
GrupoD	w54	646	114 528 0 4 0
GrupoD	w55	646	114 528 0 4 0
GrupoD	w56	646	110 532 0 4 0
GrupoD	w57	646	110 532 0 4 0

Continued on next page

Continued from previous page

Grupo	test	VG	VGTEXT
GrupoD	w58	770	110 656 0 4 0
GrupoD	w59	770	110 656 0 4 0
GrupoD	w60	770	110 656 0 4 0
GrupoD	w61	928	134 790 0 4 0
GrupoD	w62	928	134 790 0 4 0
GrupoD	w63	38	34 0 0 4 0
GrupoD	w64	27	23 0 0 4 0
GrupoD	w65	3834	286 3544 0 4 0
GrupoD	w66	4496	294 4198 0 4 0
GrupoD	w67	3834	286 3544 0 4 0
GrupoD	w68	90	86 0 0 4 0
GrupoD	w69	54	50 0 0 4 0
GrupoD	w70	38	34 0 0 4 0
GrupoD	w71	1318	133 1181 0 4 0
GrupoD	w72	1176	122 1050 0 4 0
GrupoD	w73	58	54 0 0 4 0
GrupoD	w74	1459	141 1314 0 4 0
GrupoD	w75	82	78 0 0 4 0
GrupoD	w76	81	77 0 0 4 0
GrupoD	w77	49	45 0 0 4 0
GrupoD	w78	55	51 0 0 4 0
GrupoI	w03	208	208 0 0 0 0
GrupoI	w09	208	208 0 0 0 0
GrupoI	w14	264	56 0 0 208 0
GrupoI	w68	208	208 0 0 0 0
GrupoI	w74	208	208 0 0 0 0
GrupoJ	w10	3	3 0 0 0 0
GrupoJ	w11	3	3 0 0 0 0
GrupoJ	w12	3	3 0 0 0 0
GrupoJ	w13	3	3 0 0 0 0
GrupoJ	w50	3	3 0 0 0 0
GrupoJ	w51	3	3 0 0 0 0
GrupoJ	w52	3	3 0 0 0 0
GrupoJ	w53	3	3 0 0 0 0
GrupoJ	w56	3	3 0 0 0 0
GrupoJ	w57	3	3 0 0 0 0
GrupoJ	w75	6	6 0 0 0 0
GrupoJ	w77	3	3 0 0 0 0
GrupoJ	w78	3	3 0 0 0 0
GrupoM	w01	213	210 0 0 3 0
GrupoM	w02	714	714 0 0 0 0
GrupoM	w03	510	424 84 0 2 0
GrupoM	w04	207	207 0 0 0 0
GrupoM	w05	205	205 0 0 0 0
GrupoM	w06	318	318 0 0 0 0
GrupoM	w07	152	150 0 0 2 0
GrupoM	w08	153	151 0 0 2 0
GrupoM	w09	1017	931 84 0 2 0
GrupoM	w10	380	380 0 0 0 0
GrupoM	w11	322	322 0 0 0 0
GrupoM	w12	380	380 0 0 0 0
GrupoM	w13	322	322 0 0 0 0
GrupoM	w14	153	153 0 0 0 0
GrupoM	w15	93	91 0 0 2 0
GrupoM	w16	96	94 0 0 2 0
GrupoM	w17	320	318 0 0 2 0
GrupoM	w18	489	487 0 0 2 0
GrupoM	w19	489	489 0 0 0 0
GrupoM	w20	658	656 0 0 2 0
GrupoM	w21	826	826 0 0 0 0
GrupoM	w22	321	321 0 0 0 0
GrupoM	w23	829	829 0 0 0 0
GrupoM	w24	491	489 0 0 2 0
GrupoM	w25	829	829 0 0 0 0
GrupoM	w26	1164	1164 0 0 0 0

Continued on next page

Continued from previous page

Grupo	test	VG	VGTEXT
GrupoM	w27	659	659 0 0 0 0
GrupoM	w28	205	205 0 0 0 0
GrupoM	w29	36	34 0 0 2 0
GrupoM	w30	375	375 0 0 0 0
GrupoM	w31	262	260 0 0 2 0
GrupoM	w32	321	319 0 0 2 0
GrupoM	w33	320	320 0 0 0 0
GrupoM	w34	320	318 0 0 2 0
GrupoM	w35	435	433 0 0 2 0
GrupoM	w36	435	433 0 0 2 0
GrupoM	w37	435	433 0 0 2 0
GrupoM	w38	435	433 0 0 2 0
GrupoM	w39	663	661 0 0 2 0
GrupoM	w40	262	260 0 0 2 0
GrupoM	w41	262	260 0 0 2 0
GrupoM	w42	262	260 0 0 2 0
GrupoM	w43	262	260 0 0 2 0
GrupoM	w44	262	260 0 0 2 0
GrupoM	w45	262	260 0 0 2 0
GrupoM	w46	321	319 0 0 2 0
GrupoM	w47	321	319 0 0 2 0
GrupoM	w48	321	319 0 0 2 0
GrupoM	w49	321	319 0 0 2 0
GrupoM	w50	325	323 0 0 2 0
GrupoM	w51	325	323 0 0 2 0
GrupoM	w52	325	323 0 0 2 0
GrupoM	w53	325	323 0 0 2 0
GrupoM	w54	321	319 0 0 2 0
GrupoM	w55	321	319 0 0 2 0
GrupoM	w56	325	323 0 0 2 0
GrupoM	w57	325	323 0 0 2 0
GrupoM	w58	374	374 0 0 0 0
GrupoM	w59	374	374 0 0 0 0
GrupoM	w60	374	374 0 0 0 0
GrupoM	w61	435	433 0 0 2 0
GrupoM	w62	435	433 0 0 2 0
GrupoM	w63	230	221 0 0 9 0
GrupoM	w64	207	207 0 0 0 0
GrupoM	w65	1638	1636 0 0 2 0
GrupoM	w66	1921	1919 0 0 2 0
GrupoM	w67	1638	1636 0 0 2 0
GrupoM	w68	852	768 84 0 0 0
GrupoM	w69	34	32 0 0 2 0
GrupoM	w70	382	380 0 0 2 0
GrupoM	w71	601	601 0 0 0 0
GrupoM	w72	545	543 0 0 2 0
GrupoM	w73	653	651 0 0 2 0
GrupoM	w74	666	636 28 0 2 0
GrupoM	w75	606	606 0 0 0 0
GrupoM	w76	167	108 56 0 3 0
GrupoM	w77	377	377 0 0 0 0
GrupoM	w78	489	489 0 0 0 0
GrupoO	w01	956	48 768 0 140 0
GrupoO	w02	1069	48 816 0 205 0
GrupoO	w03	1904	96 1536 0 272 0
GrupoO	w04	486	48 336 0 102 0
GrupoO	w05	534	48 384 0 102 0
GrupoO	w06	809	48 624 0 137 0
GrupoO	w07	452	48 336 0 68 0
GrupoO	w08	452	48 336 0 68 0
GrupoO	w09	2438	96 1968 0 374 0
GrupoO	w10	664	48 480 0 136 0
GrupoO	w11	809	48 624 0 137 0
GrupoO	w12	664	48 480 0 136 0
GrupoO	w13	809	48 624 0 137 0

Continued on next page

Continued from previous page

Grupo	test	VG	VGTEXT
GrupoO	w14	357	48 240 0 69 0
GrupoO	w15	274	48 192 0 34 0
GrupoO	w16	274	48 192 0 34 0
GrupoO	w17	726	48 576 0 102 0
GrupoO	w18	870	48 720 0 102 0
GrupoO	w19	904	48 720 0 136 0
GrupoO	w20	1048	48 864 0 136 0
GrupoO	w21	1164	48 912 0 204 0
GrupoO	w22	630	48 480 0 102 0
GrupoO	w23	1356	48 1104 0 204 0
GrupoO	w24	1000	48 816 0 136 0
GrupoO	w25	1390	48 1104 0 238 0
GrupoO	w26	1520	96 1152 0 272 0
GrupoO	w27	986	48 768 0 170 0
GrupoO	w28	582	48 432 0 102 0
GrupoO	w29	404	48 288 0 68 0
GrupoO	w30	712	48 528 0 136 0
GrupoO	w31	678	48 528 0 102 0
GrupoO	w32	856	48 672 0 136 0
GrupoO	w33	939	48 720 0 171 0
GrupoO	w34	939	48 720 0 171 0
GrupoO	w35	1082	48 864 0 170 0
GrupoO	w36	1082	48 864 0 170 0
GrupoO	w37	1082	48 864 0 170 0
GrupoO	w38	1082	48 864 0 170 0
GrupoO	w39	1500	48 1248 0 204 0
GrupoO	w40	678	48 528 0 102 0
GrupoO	w41	678	48 528 0 102 0
GrupoO	w42	678	48 528 0 102 0
GrupoO	w43	678	48 528 0 102 0
GrupoO	w44	678	48 528 0 102 0
GrupoO	w45	678	48 528 0 102 0
GrupoO	w46	856	48 672 0 136 0
GrupoO	w47	856	48 672 0 136 0
GrupoO	w48	856	48 672 0 136 0
GrupoO	w49	856	48 672 0 136 0
GrupoO	w50	856	48 672 0 136 0
GrupoO	w51	856	48 672 0 136 0
GrupoO	w52	856	48 672 0 136 0
GrupoO	w53	856	48 672 0 136 0
GrupoO	w54	856	48 672 0 136 0
GrupoO	w55	856	48 672 0 136 0
GrupoO	w56	856	48 672 0 136 0
GrupoO	w57	856	48 672 0 136 0
GrupoO	w58	808	48 624 0 136 0
GrupoO	w59	808	48 624 0 136 0
GrupoO	w60	808	48 624 0 136 0
GrupoO	w61	1130	48 912 0 170 0
GrupoO	w62	1130	48 912 0 170 0
GrupoO	w63	543	96 336 0 111 0
GrupoO	w64	536	96 336 0 104 0
GrupoO	w65	3712	48 3120 0 544 0
GrupoO	w66	4116	48 3456 0 612 0
GrupoO	w67	3712	48 3120 0 544 0
GrupoO	w68	2012	96 1536 0 380 0
GrupoO	w69	274	48 192 0 34 0
GrupoO	w70	713	48 528 0 137 0
GrupoO	w71	1179	48 960 0 171 0
GrupoO	w72	1082	48 864 0 170 0
GrupoO	w73	924	48 672 0 204 0
GrupoO	w74	1501	48 1248 0 205 0
GrupoO	w75	987	48 768 0 171 0
GrupoO	w76	1478	96 1104 0 278 0
GrupoO	w77	665	48 480 0 137 0
GrupoO	w78	795	48 576 0 171 0

5. Erros de geração de árvore errada

   Estes são os grupos para os quais a árvore gerada na saída não está de acordo com a árvore de referência definida de acordo com as Seções 2.3 e 2.4 da especificação da E3. O professor disponibiliza arquivos `dot` das árvores de referência (arquivos `.ref.dot`). O professor recomenda o uso da ferramenta `xdot` para visualizar as árvores (`apt install xdot`).

   ```
   xdot - interactive viewer for Graphviz dot files
   ```

   O professor também fornece as árvores geradas pelos grupos para cada uma das entradas (arquivos `.dot` em diretórios específicos para cada grupo). Assim, basta abrir a árvore de referência e a árvore gerada pelo grupo para entender o equívoco. Se dúvidas permanecerem, entre em contato com o professor.

Grupo	test
GrupoA	w10
GrupoA	w11
GrupoA	w12
GrupoA	w13
GrupoA	w37
GrupoA	w39
GrupoA	w63
GrupoA	w65
GrupoA	w67
GrupoA	w72
GrupoA	w78
GrupoB	w01
GrupoB	w02
GrupoB	w03
GrupoB	w04
GrupoB	w08
GrupoB	w09
GrupoB	w10
GrupoB	w11
GrupoB	w12
GrupoB	w13
GrupoB	w18
GrupoB	w19
GrupoB	w20
GrupoB	w21
GrupoB	w23
GrupoB	w24
GrupoB	w25
GrupoB	w26
GrupoB	w27
GrupoB	w30
GrupoB	w63
GrupoB	w64
GrupoB	w65
GrupoB	w66
GrupoB	w67
GrupoB	w68
GrupoB	w70
GrupoB	w71
GrupoB	w75
GrupoD	w01
GrupoD	w02
GrupoD	w03
GrupoD	w04
GrupoD	w05
GrupoD	w06
GrupoD	w07
GrupoD	w08
GrupoD	w09
GrupoD	w10
GrupoD	w11
GrupoD	w12
GrupoD	w13

Continued on next page

Continued from previous page

Grupo	test
GrupoD	w14
GrupoD	w15
GrupoD	w16
GrupoD	w17
GrupoD	w18
GrupoD	w19
GrupoD	w20
GrupoD	w21
GrupoD	w22
GrupoD	w23
GrupoD	w24
GrupoD	w25
GrupoD	w26
GrupoD	w27
GrupoD	w28
GrupoD	w30
GrupoD	w31
GrupoD	w32
GrupoD	w33
GrupoD	w34
GrupoD	w35
GrupoD	w36
GrupoD	w37
GrupoD	w38
GrupoD	w39
GrupoD	w40
GrupoD	w41
GrupoD	w42
GrupoD	w43
GrupoD	w44
GrupoD	w45
GrupoD	w46
GrupoD	w47
GrupoD	w48
GrupoD	w49
GrupoD	w50
GrupoD	w51
GrupoD	w52
GrupoD	w53
GrupoD	w54
GrupoD	w55
GrupoD	w56
GrupoD	w57
GrupoD	w58
GrupoD	w59
GrupoD	w60
GrupoD	w61
GrupoD	w62
GrupoD	w63
GrupoD	w64
GrupoD	w65
GrupoD	w66
GrupoD	w67
GrupoD	w68
GrupoD	w70
GrupoD	w71
GrupoD	w72
GrupoD	w73
GrupoD	w74
GrupoD	w75
GrupoD	w77
GrupoD	w78
GrupoI	w14
GrupoI	w63
GrupoJ	w03

Continued on next page

Continued from previous page

Grupo	test
GrupoJ	w09
GrupoJ	w10
GrupoJ	w11
GrupoJ	w12
GrupoJ	w13
GrupoJ	w68
GrupoJ	w74
GrupoK	w01
GrupoK	w03
GrupoK	w09
GrupoK	w29
GrupoK	w64
GrupoK	w68
GrupoK	w69
GrupoK	w75
GrupoK	w76
GrupoK	w77
GrupoK	w78
GrupoM	w00
GrupoM	w37
GrupoM	w39
GrupoM	w65
GrupoM	w67
GrupoM	w72
GrupoM	w78
GrupoN	w01
GrupoN	w02
GrupoN	w03
GrupoN	w04
GrupoN	w05
GrupoN	w06
GrupoN	w07
GrupoN	w08
GrupoN	w09
GrupoN	w10
GrupoN	w11
GrupoN	w12
GrupoN	w13
GrupoN	w14
GrupoN	w15
GrupoN	w16
GrupoN	w30
GrupoN	w31
GrupoN	w32
GrupoN	w33
GrupoN	w34
GrupoN	w35
GrupoN	w36
GrupoN	w37
GrupoN	w38
GrupoN	w39
GrupoN	w40
GrupoN	w41
GrupoN	w42
GrupoN	w43
GrupoN	w44
GrupoN	w45
GrupoN	w46
GrupoN	w47
GrupoN	w48
GrupoN	w49
GrupoN	w50
GrupoN	w51
GrupoN	w52
GrupoN	w53

Continued on next page

Continued from previous page

Grupo	test
GrupoN	w56
GrupoN	w57
GrupoN	w58
GrupoN	w59
GrupoN	w60
GrupoN	w61
GrupoN	w62
GrupoN	w64
GrupoN	w65
GrupoN	w66
GrupoN	w67
GrupoN	w70
GrupoN	w71
GrupoN	w72
GrupoN	w73
GrupoN	w74
GrupoN	w76
GrupoO	w00
GrupoO	w01
GrupoO	w02
GrupoO	w03
GrupoO	w04
GrupoO	w05
GrupoO	w06
GrupoO	w07
GrupoO	w08
GrupoO	w09
GrupoO	w10
GrupoO	w11
GrupoO	w12
GrupoO	w13
GrupoO	w14
GrupoO	w15
GrupoO	w16
GrupoO	w17
GrupoO	w18
GrupoO	w19
GrupoO	w20
GrupoO	w21
GrupoO	w22
GrupoO	w23
GrupoO	w24
GrupoO	w25
GrupoO	w26
GrupoO	w27
GrupoO	w28
GrupoO	w29
GrupoO	w30
GrupoO	w31
GrupoO	w32
GrupoO	w33
GrupoO	w34
GrupoO	w35
GrupoO	w36
GrupoO	w37
GrupoO	w38
GrupoO	w39
GrupoO	w40
GrupoO	w41
GrupoO	w42
GrupoO	w43
GrupoO	w44
GrupoO	w45
GrupoO	w46
GrupoO	w47

Continued on next page

Continued from previous page

Grupo	test
GrupoO	w48
GrupoO	w49
GrupoO	w50
GrupoO	w51
GrupoO	w52
GrupoO	w53
GrupoO	w54
GrupoO	w55
GrupoO	w56
GrupoO	w57
GrupoO	w58
GrupoO	w59
GrupoO	w60
GrupoO	w61
GrupoO	w62
GrupoO	w63
GrupoO	w64
GrupoO	w65
GrupoO	w66
GrupoO	w67
GrupoO	w68
GrupoO	w69
GrupoO	w70
GrupoO	w71
GrupoO	w72
GrupoO	w73
GrupoO	w74
GrupoO	w75
GrupoO	w76
GrupoO	w77
GrupoO	w78
GrupoP	w06
GrupoP	w11
GrupoP	w13
GrupoP	w33
GrupoP	w34
GrupoP	w71
GrupoP	w73
GrupoP	w74

1.2.3 Tabela completa

A tabela completa consiste em uma tabela com 10 colunas. As colunas são:

- Grupo: o identificador do grupo
- test: o identificador do teste (veja o `tgz` com os arquivos)
- TCNO: valor de retorno do programa que compara a árvore
- VG: somatório dos valores obtidos do valgrind
- TCNOTEXT: saída padrão/erro do programa do grupo
- VGTEXT: números reportados pelo valgrind, detalhados
- Output: saída do script que tenta executar o programa do grupo
- Decision.TCNO: TRUE, teste acertou a saída (árvore idêntica); FALSE, teste falhou (há alguma diferença na árvore)
- Decision.VG: TRUE, quando VG == 0; FALSE, caso contrário
- Decision: decisão final se o teste passou ou não. De acordo com a especificação E3, para o teste ser aceito, (Decision.TCNO deve ser TRUE && Decision.VG deve ser também TRUE)

```
read_csv("e3/e3_output_objetivo_completa.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 1,185 × 10
   Grupo  test   TCNO    VG TCNOTEXT VGTEXT     Output Decision.TCNO Decision.VG
   <chr>  <chr> <dbl> <dbl> <lgl>    <chr>      <chr>  <lgl>         <lgl>
 1 GrupoA w00       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 2 GrupoA w01       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 3 GrupoA w02       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 4 GrupoA w03       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 5 GrupoA w04       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 6 GrupoA w05       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 7 GrupoA w06       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 8 GrupoA w07       0     0 NA       no leaks ... <NA>   TRUE          TRUE
 9 GrupoA w08       0     0 NA       no leaks ... <NA>   TRUE          TRUE
10 GrupoA w09       0     0 NA       no leaks ... <NA>   TRUE          TRUE
# ... with 1,175 more rows, and 1 more variable: Decision <lgl>
```

1.2.4 Tabela simplificada

A tabela simplificada consiste em uma tabela com 5 colunas. As colunas são:

- Grupo: o identificador do grupo
- test: o identificador do teste (veja o `tgz` com os arquivos)
- TCNO: valor de retorno do programa que compara a árvore
- VG: somatório dos valores obtidos do valgrind
- Decision: decisão final se o teste passou ou não. De acordo com a especificação E3, para o teste ser aceito, (Decision.TCNO deve ser TRUE && Decision.VG deve ser também TRUE)

```
read_csv("e3/e3_output_objetivo.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 1,185 × 5
   Grupo  test   TCNO    VG Decision
   <chr>  <chr> <dbl> <dbl> <lgl>
 1 GrupoA w00       0     0 TRUE
 2 GrupoA w01       0     0 TRUE
 3 GrupoA w02       0     0 TRUE
 4 GrupoA w03       0     0 TRUE
 5 GrupoA w04       0     0 TRUE
 6 GrupoA w05       0     0 TRUE
 7 GrupoA w06       0     0 TRUE
 8 GrupoA w07       0     0 TRUE
 9 GrupoA w08       0     0 TRUE
10 GrupoA w09       0     0 TRUE
# ... with 1,175 more rows
```

1.2.5 Tabela com saída

Esta tabela mostra a saída, normalmente indica o erro.

```
read_csv("e3/e3_output_errors.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 183 × 3
   Grupo  test  Output
   <chr>  <chr> <chr>
 1 GrupoA w22   line 4: syntax error, unexpected TOKEN_ERRO
 2 GrupoB w55   Segmentation fault
 3 GrupoC w68   line 3 - syntax error, unexpected TK_LIT_INT
 4 GrupoG w00   timeout: failed to run command `./etapa3': No such file or dire...
 5 GrupoG w01   timeout: failed to run command `./etapa3': No such file or dire...
 6 GrupoG w02   timeout: failed to run command `./etapa3': No such file or dire...
 7 GrupoG w03   timeout: failed to run command `./etapa3': No such file or dire...
 8 GrupoG w04   timeout: failed to run command `./etapa3': No such file or dire...
 9 GrupoG w05   timeout: failed to run command `./etapa3': No such file or dire...
10 GrupoG w06   timeout: failed to run command `./etapa3': No such file or dire...
# ... with 173 more rows
```

1.2.6 Tabela de resultados com vazamento de memória

Esta tabela mostra a quantidade de bytes (na coluna VG) que não haviam sido liberados no final do programa, obtido através do valgrind.

```
read_csv("e3/e3_leaks.csv", col_types=cols(), progress=FALSE)
```

```
# A tibble: 329 × 3
   Grupo  test     VG
   <chr>  <chr> <dbl>
 1 GrupoB w01     120
 2 GrupoB w02     146
 3 GrupoB w03     254
 4 GrupoB w04     280
 5 GrupoB w05      18
 6 GrupoB w06     179
 7 GrupoB w07      24
 8 GrupoB w08     114
 9 GrupoB w09     380
10 GrupoB w10     330
# ... with 319 more rows
```

1.3 Pesos

Grupo	E3.P
GrupoA	0.8
GrupoB	1
GrupoC	1
GrupoD	0.8
GrupoE	1
GrupoG	1
GrupoH	1
GrupoI	1
GrupoJ	1
GrupoK	1
GrupoM	1
GrupoN	1
GrupoO	0.8
GrupoP	1
GrupoT	1

1.4 Final

Grupo	Etapa	E3.O	E3.S	E3.P
GrupoA	E3	8.48	8	0.8
GrupoB	E3	0.13	7.3	1
GrupoC	E3	9.87	9.35	1
GrupoD	E3	0.13	6	0.8
GrupoE	E3	10	9.9	1
GrupoG	E3	0	0	1
GrupoH	E3	10	9.8	1
GrupoI	E3	8.99	8.8	1
GrupoJ	E3	7.85	9	1
GrupoK	E3	8.61	9.25	1
GrupoM	E3	0	8.75	1
GrupoN	E3	0.13	7	1
GrupoO	E3	0	7.4	0.8
GrupoP	E3	8.99	9.5	1
GrupoT	E3	0.13	6.3	1

Grupos em recuperação da E3:

- **Apenas grupos que entregaram no prazo** conforme regramentos
- Em Moodle já configurado para tal, use o link "Recuperação E3"
- Política de recuperação ativada (vejam regras gerais)

```
[1] "GrupoB" "GrupoG" "GrupoM" "GrupoN" "GrupoT"
```