Especificação e Verificação de Programas

Trabalho 1 - Especificação e Verificação Formal

O trabalho pode ser realizado individualmente, ou em equipes com até DOIS alunos.

A linguagem de especificação utilizada nas questões é o JML das atribuições é Java. Para a resolução das questões abaixo sugerimos a utilização do OpenJML, disponível para download em http://www.openjml.org ou online em um dos seguintes sites:

- [Site da Chalmers] http://cse-212294.cse.chalmers.se/courses/sefm/openjml/
- [site oficial] https://www.rise4fun.com/OpenJMLESC/

As questões de verificação formal usam a lógica Hoare com updates sobre a linguagem While, conforme apresentado em sala de aula. Você deve usar a ferramenta KeY-Hoare para verificar os programas formalmente. Para fazer o download e ter instruções de como começar, veja o link e o artigo abaixo:

- http://www.key-project.org/download/hoare/
- http://www.key-project.org/download/hoare/students.pdf

Sempre que for solicitado verificar um programa, construa uma prova à mão (sem utilizar o sistema KeY-Hoare). Inicialmente, defina a tripla de Hoare triple com atualizações que represente o problema. Em seguida, em cada etapa da prova aplique uma regra Hoare, indicando qual regra foi aplicada e também o(s) seu(s) resultado (um ou vários novas triplas de Hoare com atualizações, ou uma fórmula lógica).

- Se o aplicação da regra resultar em uma nova tripla de Hoare, simplesmente continue com a prova da mesma maneira;
- Se resultar em uma fórmula lógica, explique por que a fórmula é válida.
- Se a aplicação da regra resultar em várias novos constraints (Hoare triple ou formula), escolha uma delas e continue a comprovar este constraints. Quando o primeiro constraint tiver sido provado, retorne aos demais prove eles também. Numere os resultados intermediários (e ramificações de prova) para que não haja confusão.
- Se os programas contiverem várias atribuições em uma linha, você poderá aplicar a regra de atribuição para todos eles antes de apresentar a tripla de Hoare resultante.
- Se você usou a regra de invariante de loop, indique claramente qual fórmula você está usando como invariante.

As provas podem ser bastante longas com várias triplas similares de Hoare. Você pode copiar e colar para evitar a digitação excessiva. Quando você está lidando com grandes fórmulas na pré ou pós-condição, você pode introduzir nomes para abreviá-las.

Sempre que lhe for pedido para provar algo usando o KeY-Hoare, você deve:

- Escreva um arquivo .key correspondente ao triplo Hoare.
- Carregue o arquivo no KeY-Hoare e faça a prova.
- Por fim, salve a prova em um arquivo .key.proof.

NOTA IMPORTANTE: Se a sua prova não fechar, envie o arquivo .proof não fechado de qualquer maneira! Desta forma, podemos avaliar a sua evolução.

1. Especificando Implementação [3 pontos]

Considere a classe SortedIntegers, que é semelhante à uma classe mostrada em sala. Um objeto SortedIntegers pode conter elementos inteiros duplicados.

Observe que o array arr deve estar ordenado o tempo todo e as implementações dos vários métodos dependerão disso.

```
public class SortedIntegers {
        private int arr[];
        private int capacity, size = 0;

        public SortedIntegers(int capacity) {
           this.capacity = capacity;
           this.arr = new int[capacity];
        }
        public void add(int elem) {
           // ...
        }
        public void remove(int elem) {
           // ...
        }
        public boolean contains(int elem) {
           // ...
        }
        public int max() {
           // ...
        }
        public int getSize() {
           // ...
        }
        public int getCapacity() {
           // ...
        }
        public String toString() {
           // ...
        }
    }
```

a) Especificação de SortedIntegers: Especifique todos os métodos em SortedIntegers (exceto toString ()) usando JML.

b) Invariantes: Você usou invariantes em sua especificação? Se você não fez isso, volte ao ponto anterior e especifique usando invariantes. Explique no relatório por que é útil usar invariantes; Ao explicá-lo, dê um exemplo concreto usando a especificação do método de adicionar ou remover.

c) Implementação de SortedIntegers: Implemente os métodos em SortedIntegers, cumprindo

integralmente com a especificação.

2. Da especificando à implementação e verificação [3 pontos]

Um programa chamado isorttriple tem a seguinte especificação:

```
requires: a <= b & a = _a & b = _b & c = _c
ensures: a <= b & b <= c &
        (a = _a & b = _b & c = _c |
         a = _a & b = _c & c = _b |
         a = _b & b = _a & c = _c |
         a = _b & b = _c & c = _a |
         a = _c & b = _a & c = _b |
         a = _c & b = _b & c = _a)
```

_a,_b and _c SÃO constantes lógicas. Todos locations e termos são inteiros.

a) Dê uma implementação de isorttriple.
b) Prove manualmente que sua implementação está correta.
c) Prove a mesma coisa em KeY-Hoare.

3. Especificação e Verificação [4 pontos]

O seguinte trecho de código multiplica dois inteiros.

```
res = 0;
 if (n < 0) {
   n = -n;
   m = -m;
 } else {
 }
 while (n>0) {
   res = res + m;
   n = n - 1;
 }
```

n e m são as duas entradas inteiras para o programa. res é o resultado inteiro. Observe que você deve aplicar a regra de loop duas vezes (embora haja um loop único), uma vez após a ramificação if-then ter sido executada e uma vez após a ramificação else ter sido executada.

a) Qual deve ser a especificação do programa?
    i) Obs.: Apesar de você poder sempre dar uma especificação igênua para a qual o programa está correto, isso não será considerado uma especificação correta.
b) Prove manualmente que o programa está correto (corretude total) com relação à sua especificação. Existem dois casos semelhantes na prova, um para cada invariante de loop. Você só precisa incluir um desses casos na prova feita à mão.

c) Prove o seu programa no KeY-Hoare (considerando todas as ramificações do condicional).

Relatórios
Envie um arquivo TRAB2.zip (ou rar, tar.gz, tar como desejar) usando o SIGAA, contendo os seguintes arquivos:

- SortedIntegers.java contendo a implementação e especificação da questão 1
- isorttriple.txt - resposta da questão 2a e 2b
- isorttriple.key and isorttriple.key.proof - resposta da questão 2c
- multiply.txt - resposta da questão 3a e 3b
- multiply.key and multiply.key.proof - resposta da questão 3c

Observação: não precisa ser mais complexo do que isso; evite adicionar arquivos desnecessários, como arquivos ocultos ou pastas produzidos por seu editor de texto, por exemplo.