DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Ingeniería de software II

Trabajo Práctico N°4

Integrante	LU	Correo electrónico
Esteban Luciano Rey	657/10	estebanlucianorey@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Índice

1. Respuestas

1.1. Ejercicio 1

a)

```
(declare-const x Bool)
(declare-const y Bool)
(assert (= (not (or x y)) (and (not x) (not y) )))
(check-sat)
(get-model)
```

b)

```
(declare-const x Bool)
(declare-const y Bool)
(assert ( = (and x y) (not (or (not x)(not y)))))
(check-sat)
(get-model)
```

c)

```
(declare-const x Bool)
(declare-const y Bool)
(assert ( = (not (and x y)) (not (and (not x)(not y)))))
(check-sat)
(get-model)
```

1.2. Ejercicio 2

a)

```
(declare-fun x () Int)
(declare-fun y () Int)
(assert (= (+ (* x 3) (* y 2) ) 36))
(check-sat)
(get-model)
```

satisface la formula con x = 0 e y = 12

b)

```
(declare-fun x () Int)
(declare-fun y () Int)
(assert (= (+ (* x 5) (* y 4) ) 64))
(check-sat)
(get-model)
```

satisface la formula con x = 1 e y = 12

c)

```
(declare-fun x () Int)
(declare-fun y () Int)
(assert (= (* x y ) 64))
(check-sat)
(get-model)
```

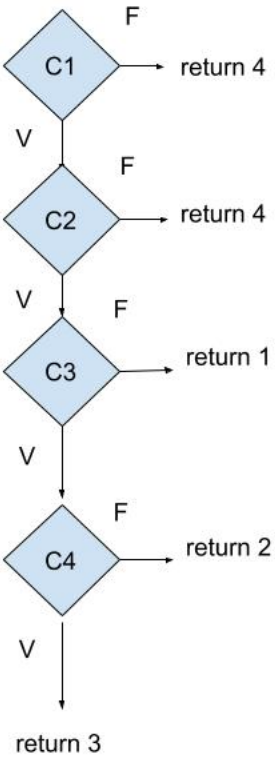
Satisface la formula con x = 1 e y = 64

1.3. Ejercicio 3

```
(declare-const a1 Real)
(declare-const a2 Real)
(declare-const a3 Real)
(assert (= a1 (mod 16 2)))
(assert (= a2 (/ 16 4)))
(assert (= a3 (div 16 5)))
(check-sat)
(get-model)
```

1.4. Ejercicio 4

a)



b)

JaCoCo reporta una cobertura del % 85 en donde reporta que no se visitaron las branches de equilateral ni isoceles. Esto se da por la naturaleza random de generación de tests de Randoop: no llegó a generar en esos 2 segundos los testeos necesarios para cubrir esas lineas.

c)

$$C1 = (a_0 \leq 0 \vee b_0 \leq 0 \vee c_0 \leq 0)$$
$$C2 = \neg(a+b > c \wedge a+c > b \wedge b+c > a)$$
$$C3 = (a = b \wedge b = c)$$
$$C4 = (a = b \vee b = c \vee a = c)$$

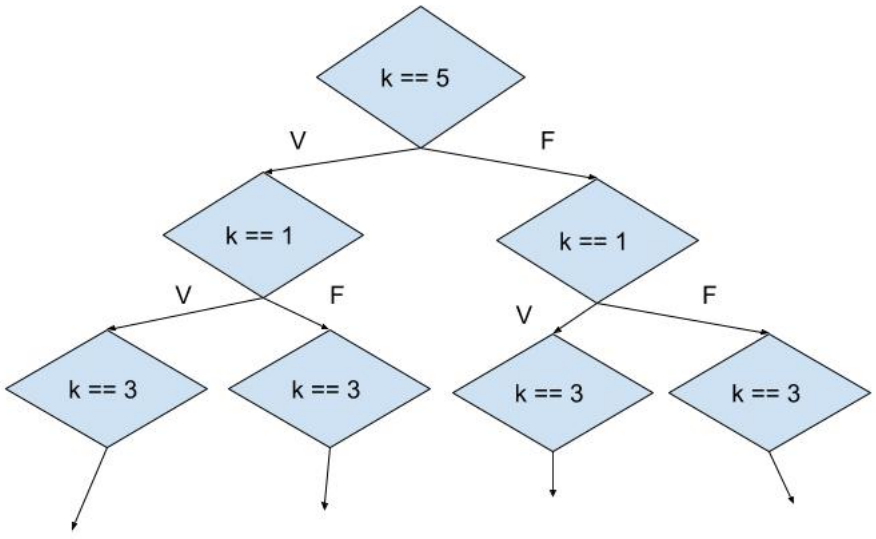
#iter	Input concreto (a,b,c)	Condición de ruta	Especificaciones para Z3	Resultado Z3
1	(0, 0, 0)	C1	(assert (not (or (<= a 0) (<= b 0) (<= c 0))))	(1, 1, 1)
2	(1, 1, 1)	¬C1 ∧ ¬C2 ∧ C3	(assert (not (or (<= a 0) (<= b 0) (<= c 0)))) (assert (and (> (a b) c) (> (a c) b) (> (b c) a))) (assert (not (and (= a b) (= b c))))	(2, 3, 4)
3	(2, 3, 4)	¬C1 ∧ ¬C2 ∧ ¬C3 ∧ ¬C4	(assert (not (or (<= a 0) (<= b 0) (<= c 0)))) (assert (and (> (a b) c) (> (a c) b) (> (b c) a))) (assert (not (and (= a b) (= b c)))) (assert (or (= a b) (= b c) (= a c)))	(2, 1, 2)
4	(2, 1, 2)	¬C1 ∧ ¬C2 ∧ ¬C3 ∧ C4	(assert (not (or (<= a 0) (<= b 0) (<= c 0)))) (assert (not (and (> (a b) c) (> (+ a c) b) (> (b c) a))))	(1, 1, 2)
5	(1, 1, 2)	¬C1 ∧ C2	END	END

d)

Cobertura del %100

1.5. Ejercicio 5

a)



b)

JaCoCo reporta una cobertura del %100. Que se haya llegado de forma aleatorea a probar justo por alguno de los 3 valores de double que cumplen la condición del if es muy poco probable, por lo que revisando los tests generados, se puede ver que uno de los valores con los que prueba Randoop es el -1 en distintos formatos (casteado como Long, como float...etc), con lo cual, las 2 ramas del if dentro del for son probadas.

c)

Las ramas que no pueden ser satisfechas no se informan (por ejemplo $k\bar{5} \wedge k\bar{1} \wedge k\bar{3}$)

$$C1 = (k = 5)$$
$$C2 = (k = 1)$$
$$C3 = (k = 3)$$

#iter	Input concreto k	Condición de ruta	Especificaciones para Z3	Resultado Z3
1	0	¬C1 ∧ ¬C2 ∧ ¬C3	(assert (not (= k 5))) (assert (not (= k 1))) (assert (= k 3))	3
2	3	¬C1 ∧ ¬C2 ∧ C3	(assert (= k 5)) (assert (not (= k 1))) (assert (not (= k 1)))	5
3	5	C1 ∧ ¬C2 ∧ ¬C3	(assert (not (= k 5))) (assert (= k 1))) (assert (not (= k 1)))	1
4	1	¬C1 ∧ ¬C2 ∧ C3	END	END

d)

Cobertura del %100 de las lineas de codigo, en cuanto al arbol de computo hay ramas que son imposibles de alcanzar