CS 2209 B
Assignment 4

Student Name : Ashna Mittal
Student ID : 251206758

Exercise 1:-

```
q := 0
r := x;
while (r >= y)
  {
      r := r - y;
      q := q + 1;
  }
```

The above program calculates the quotient and remainder when y is ~~is~~ the divisor and x is the dividend, i.e., x is divided by y. (x/y). It first initializes the variables q(quotient) to 0 and r(remainder) to x. It then runs a while loop till r, the ~~dividend~~ remainder, is greater than or equal to the divisor, y. The loop would only terminate when r< y. Inside the loop, in each iteration, the value of r is decremented by the value of y and q is incremented by 1.
For example, let x = 24, y = 5.
Here, r = 24 and when the while loop is entered:

1st Iteration :- (24 >= 5)

r = 24 - 5 = 19

q = 0+1 = 1

2nd iteration :- (19 >= 5)

r = 19 - 5 = 14

q = 1+1 = 2

3rd iteration :- (14 >= 5)

r = 14 - 5 = 9

q = 2 + 1 = 3

4th iteration :- (9 >= 5)

r = 9 - 5 = 4

q = 3 + 1 = 4

Now, the while loop terminates since in the ~~5rd~~ 5th iteration, r = 4 is less than y = 5. Hence 24 15 gives quotient as 4 and remainder = 4. The preconditions in this case were : x = 24 and y = 5.

The post conditions were : x = 24, y = 5, r = 4 and q = 4.

In General, the Program P have the following preconditions ϕ :- x := some value, y := some value ~~greater than~~ not equal to 0

The Program P will have the following post conditions : x := some value assigned initially, y := some value assigned initially, r = some value calculated, q := some value calculated.

Pre - condition (ϕ) = (x >= 0 ∧ y > 0)

Post - condition (ψ) = q = L x/y J

Exercise 2 :-

- $(x = a)\ y := x\ ;\ (y = a)$

This Hoare sentence is correct for the domain of variables in the $\mathbb{Z}$ Integer set. We assume x to be an integer value 'a' and then assign y to 'x' which means that now y is that integer value 'a'. Hence, the precondiction $x = a$ holds and the postcondition $y = a$. is satisfied after operation 'P' : $y = x$ is executed. Thus, the pre condition always satisfies the post-~~condion~~ condition. For Example, Let $x = -2$, Here, $a = -2$. Then $y := x$ gives $y = -2$ which means $y := a$ $(y != -2)$. Hence, this sentence is correct.

- $(x = a \wedge y = b)\ z := x\ ;\ x := y\ ;\ y := z\ ;\ (x = b \wedge y = a)$

This Hoare sentence is CORRECT for the domain of variables in the $\mathbb{Z}$ integer set. The precondition states that x is equal to an integer 'a' and y is equal to an integer 'b'. After program 'P' executes, 'z' is used to swap the values of 'x' and 'y' which results in the post condition where $x = b$ and $y = a$. Hence, the pre-condition always satisfies the post condition. For example ($x = -1$ and $y = 2$). $z = -1$; $x = 2$; $y = -1$. Then, we get ($x = 2$ and $y = -1$) Hence, this sentence is correct.

- $(x=a \wedge y=b)$ while $(y>0)\{x:=x+1; y:=y-1\}(x=a+b)$.

This Hoare Sentence is PARTIALLY CORRECT for the domain of variables in the $\mathbb{Z}$ integer set. The pre-condition assigns an integer value 'a' to x and 'b' to y. Program 'P' executes a while loop which runs till the value of y is greater than 0. Inside the loop, x is incremented by 1 and y is decremented by 1 in each iteration. When the program terminates, the post condition is that x is the sum of a and b. However, this postcondition is only a result if 'b' is a positive integer value. Then the while loop executes 'b' times. Hence, 'x' is increamented 'b' times and since 'x' was assigned 'a', x will be 'a+b' after the program terminates. For example, $(x=1 \wedge y=2)$ ~~while 2~~ iteration 1 : while $(2>0)\{x=1+1; y=2-1;\}$ Iteration 2 : while $(1>0)\{x=2+1; y=1-1;\}$. This results in $x=3$ which is $a+b = 1+2 = 3$. However, the postcondition is not ~~true~~ a result if 'b' is a negative integer. Since, the while loop would never execute in that scenario and the program would terminate without any changes in any values resulting in $x=a$ post condition, not $x=a+b$. For example, Let ($x=3$ and $y=-2$) while $(-2>0)$ is false, the loop doesn't execute. Hence post condition $x = 3+(2) = 1$ doesn't hold, since $x=-2$ is uncharged. Thus, the post condition is not always satisfied even though the pre-conditions hold. Therefore, the statement is partially correct.

- $(y>0)\ x:=1;\ \text{while}\ (y>0)\{\ x:=2*x;\ y:=y-1;\ \}\ (x=2^y)$

This Hoare Sentence is INCORRECT for the domain of variables in the $\mathbb{Z}$ integer set. The pre-condition assigns y a value greater than 0 i.e, a positive value. When program 'P' executes, x is assigned 1 initially and while loop is executed till y is greater than 0. Inside the loop, x is doubled in each iteration and y is decreamented by 1. The resulting post condition is $x = 2^y$ which is not true since at the end of the program y will always be equal to zero in order to terminate the loop and so will always lead to x being equal to 1, whereas, x will have a value as calculated inside the loop by doubling in each iteration. For example, let $y = 2$ satisfy the precondition: $(2>0)$. When program 'P' runs,

while $(2>0)\{\ x=2*1;\ y=2-1;\ \}$ — 1st iteration.

while $(1>0)\{\ x=2*2;\ y=1-1;\ \}$ — 2nd iteration

The resulting values are: $y = 0$ and $x = 4$.

However, the given post-condition states that $x = 2^y$ which $x = 2^0 = 1 \neq 4$. Hence, the Hoare sentence is uncorrect in all cases even when the preconditions are satisfied, but the postcondition remains unsatisfied.

Exercise 3 :-

```
1 father (terach, abraham).
2 father (terach, nachor).
3 father (abraham, isaac).
4 mother (sara, isaac).
5 parent(X, Y) :- father (X, Y).
6 parent (X, Y) :- mother (X, Y).
7 ancestor (X, Y) :- ancestor (X, Z), parent (Z, Y).
8 ancestor (X, Y) :- parent (X, Y).
  ?- ancestor (X, Y).
```

The above code generates the output:

Stack limit (0.2Gb) exceeded
Stack sizes: Local: 0.2Gb, global: 11.3Mb, trail: 1Kb
Stack depth: 1,475,928, Last-call: 0%, Choice points: 1,475,908
Probable infinite recursion (cycle):
[1,475,928]: ancestor (_1650, _1652)
[1,475,927] ancestor (_1676, _1678)

The prolog generates this error because ancestor (X, Y) keeps calling itself indefinitely because it is defined recursively without a base case. This results in a 'stack limit exceeded' exception. Here, in the line 7, ancestor (X, Y) keeps calling itself → ancestor(X, Y) → ancestor (X, Z), parent (X, Z) → ancestor (X, U), parent (U, Z), parent (Z, Y) and so on, resulting in an infinite recursive call cycle

By replacing line 7 with:
ancestor(X,Y) :- parent(X,Y), ancestor(Y,X)
we get the following output :-

X = terach,
Y = abraham,
X = terach,
Y = nachor,
X = abraham,
Y = isaac,
X = sara,
Y = isaac

We are able to resolve the previous error and get this result because after the replacement of line 7, we are first calling parent(X,Y) and after that ancestor(Y,X). Hence, the prolog will fetch false for ancestor(Y,X) being true and so ancestor(X,Y) can't be true. Then the control comes to line 8, where the result will be given for all possible outcomes. So, in our 1st iteration, ancestor(X,Y) will be called first and since we call parent(X,Y) then first, we do not get stuck in the infinite recursive loop of the ancestor(X,Y), giving us a finite answer stated above. The prolog gets all parents but not all ancestors (which is why x = terach, y = isaac is absent from result). When Parent(X,Y) is called, it calls ancestor(Y,X) which further calls parent(Y,X). If parent(X,Y) is true, then parent(Y,X) is false and the prolog terminates. Thus, the result only consists of parents and not ancestors.