Name: NIM:

Problem Set 3
TK2ICM: *Logic Programming* (CSH4Y3)
Second Term 2019-2020

Day, date	:	Monday, February 24, 2020
Duration	:	**75 minutes**
Type	:	***open all***, *individual (no cooperation between/among class participants)*

Instruction:

1. You are not allowed to discuss these problems with other class participants.
2. You may use any reference (books, slides, internet) as well as other students who are not enrolled to this class.
3. Use the predicate name as described in each of the problem. **The name of the predicate must be precisely identical**. Typographical error may lead to the cancellation of your points.
4. Submit your work to the provided slot at LMS under the file name `PS3-<iGracias_ID>.pl`. For example: `PS3-albert.pl` if your iGracias ID is `albert`.

1 Modular Addition and Multiplication

Remark 1 This problem is worth **20 points**.

In number theory, we have learnt about modular addition and multiplication. Recall that the ring of integer modulo n for an integer $n \geq 2$ is defined as the set $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$ equipped with the following addition and multiplication operations:

$$a + b \stackrel{\text{def}}{=} (a + b) \bmod n, \text{ and}$$
$$a \cdot b \stackrel{\text{def}}{=} (a \cdot b) \bmod n.$$

For examples:

- In $\mathbb{Z}_4 = \{0, 1, 2, 3\}$, we have $2 + 3 \stackrel{\text{def}}{=} (2 + 3) \bmod 4 = 5 \bmod 4 = 1$ and $2 \cdot 3 \stackrel{\text{def}}{=} (2 \cdot 3) \bmod 4 = 6 \bmod 4 = 2$.
- In $\mathbb{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$, we have $3 + 6 \stackrel{\text{def}}{=} (3 + 6) \bmod 7 = 9 \bmod 7 = 2$ and $3 \cdot 6 \stackrel{\text{def}}{=} (3 \cdot 6) \bmod 7 = 18 \bmod 7 = 4$.

The modular addition and multiplication in $\mathbb{Z}_n$ can also be extended to the set of all positive integers. Given two non-negative integers a and b and an integer $n \geq 2$, we define

$$a +_n b \stackrel{\text{def}}{=} (a + b) \bmod n, \text{ and}$$
$$a \cdot_n b \stackrel{\text{def}}{=} (a \cdot b) \bmod n.$$

From this definition, for example, we have:

- $9 +_{11} 14 = (9 + 14) \bmod 11 = 23 \bmod 11 = 1$.
- $9 \cdot_{11} 14 = (9 \cdot 14) \bmod 11 = 126 \bmod 11 = 5$.

In this problem, your tasks are as follows:

(a). **[10 points]** Write the predicate `modadd/3`, `modadd(+A,+B,+N,Result)` succeeds whenever `Result` is equal to $(\texttt{A} + \texttt{B}) \bmod \texttt{N}$, for example:

- `modadd(2,3,4,Result).` returns `Result = 1`,
- `modadd(3,6,7,Result).` returns `Result = 2`,
- `modadd(2^60+3,2^60+17,1001,Result).` returns `Result = 22`,
- `modadd(2^100+15,2^100+23,10^9+9,Result).` returns `Result = 672572594`,
- `modadd(10^100+19,10^100+21,10^9+9,Result).` returns `Result = 378813512`.

(b). **[10 points]** Write the predicate `modmul/3`, `modmul(+A,+B,+N,Result)` succeeds whenever `Result` is equal to $(\texttt{A} \cdot \texttt{B}) \bmod \texttt{N}$, for example:

- `modmul(2,3,4,Result).` returns `Result = 2`,
- `modmul(3,6,7,Result).` returns `Result = 4`,
- `modmul(2^60+3,2^60+17,1001,Result).` returns `Result = 72`,
- `modmul(2^100+15,2^100+23,10^9+9,Result).` returns `Result = 532175945`,
- `modmul(10^100+19,10^100+21,10^9+9,Result).` returns `Result = 895569273`.

Note: the above examples have been tested using SWI-Prolog 8.0.3 under Windows 7 operating system.

2 Counting the Number of Permutations

Remark 2 This problem is worth **20 points**.

Given a set $A = \{1, 2, 3, \ldots, n\}$ of n positive integers and an integer $0 \leq k \leq n$, a k-permutation of the elements of A is an ordered arrangement of k elements of A. Some examples of 3-permutation of $A = \{1, 2, 3, 4, 5\}$ are:

$$(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 5), (2, 5, 3).$$

Here $(1, 2, 3) \neq (1, 3, 2) \neq (2, 1, 3)$ because the order of the elements are different. From discrete mathematics, we know that the number of k-permutation of n elements is given by the formula

$$P(n, r) = \frac{n!}{(n-r)!},$$

where ! denotes the factorial function (see Problem Set 2).

In this problem, your task is to define the predicate `perm(+N,+R,Result)` that succeeds whenever $P(N, R) = \frac{N!}{(N-R)!} = Result$ for a non-negative integer N and R such that $0 \leq R \leq N$. Since the result can be quite large, we reduce it modulo $10^9 + 7$. This problem assumes that $0 \leq N, R \leq 10^5$. In addition, the predicate `perm/3` returns `false` if $R > N$ or if the values of N or R are negative. The first and the second argument (i.e., `+N` and `+R`) are always instantiated.

Some examples are:

- `perm(10,8,R).` returns `R = 1814400`, this means $P(10, 8) \bmod (10^9 + 7) = 1814400$;
- `perm(100,80,R).` returns `R = 257091983`, this means $P(100, 80) \bmod (10^9 + 7) = 257091983$;
- `perm(80000,79000,R).` returns `R = 144952058`, this means $P(80000, 79000) \bmod (10^9 + 7) = 144952058$;
- `perm(100000,99999,R).` returns `R = 457992974`, this means $P(100000, 99999) \bmod (10^9 + 7) = 457992974$;
- `perm(9,10,R).` returns **false** since $P(9, 10)$ is undefined (because $9 < 10$).
- `perm(9,-1,R).` returns **false** since $P(9, -1)$ is undefined (because $-1 < 0$).
- `perm(-1,0,R).` returns **false** since $P(-1, 0)$ is undefined (because $-1 < 0$).

Note: the above examples have been tested using SWI-Prolog 8.0.3 under Windows 7 operating system. The result should be obtained in under 10 seconds.

Hint 1 Express $P(n, r)$ as a product of $r + 1$ consecuive number and use the result in Problem 1.

3 Fibonacci Sequence

Remark 3 This problem is worth **20 points**.

Every undergraduate student of Informatics at School of Computing in Telkom University must learn recurrence relation (or recursive relation) in Discrete Mathematics A (this course is offered annually every second term in the first year). Recurrence relation has a profound usefulness in algorithm analysis. This relation deals with functions that are defined *recursively*, i.e., their definitions refer to themselves. One of the famous recurrence relation is defined to generate Fibonacci sequence and it is defined as follows

$$F_n = F_{n-1} + F_{n-2}, \tag{1}$$

i.e., to compute the n-th term of this sequence, we add two previous consecutive terms. This recursive computation requires one to define the value of F_0 and F_1, and normally we define $F_0 = 0$ and $F_1 = 1$. These values of F_0 and F_1 are usually called as *initial condition* (or *seeds*). For $F_0 = 0$ and $F_1 = 1$, we have following Fibonacci sequence: $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$. In this problem your task is to compute the n-th term of a Fibonacci sequence with particular initial conditions (not necessarily $F_0 = 0$ and $F_1 = 1$).

This problem ask you to write a predicate `fibo/4`, `fibo(+F0,+F1,N,FN)` succeeds whenever the first two arguments are instantiated, the third argument is a non-negative integer N, and the last argument is the N-th term of a Fibonacci sequence whose initial seeds are `+F0` and `+F1`. *You may assume that the first two arguments are always instantiated*. To make the program efficient, the value of F_0 and F_1 are restricted to numbers between -10 and 10 (inclusive), while the value of N is restricted to a non-negative integer between 0 and 20 (inclusive).

Some test cases:

- `?- fibo(0,1,7,F).` returns `F = 13`. This means, for the Fibonacci sequence whose initial condition is $F_0 = 0$ and $F_1 = 1$, we have $F_7 = 13$.
- `?- fibo(1,1,10,F).` returns `F = 89`. This means, for the Fibonacci sequence whose initial condition is $F_0 = 1$ and $F_1 = 1$, we have $F_{10} = 89$.
- `?- fibo(10,10,20,F).` returns `F = 109460`. This means, for the Fibonacci sequence whose initial condition is $F_0 = 10$ and $F_1 = 10$, we have $F_{20} = 109\,460$.
- `?- fibo(-9,-10,20,F).` returns `F = -105279`. This means, for the Fibonacci sequence whose initial condition is $F_0 = -9$ and $F_1 = -10$, we have $F_{20} = -105\,279$.
- `?- fibo(1,2,-2,F).` returns **false**, since the third argument is a negative integer.

Side effects such as **true** or **false** are admissible. However, your program should avoid infinite recursive call.

4 Triangular Pattern

Remark 4 This problem is worth **20 points**.

4.1 Inverted Triangular Pattern

(This subproblem is worth **10 points**.)

Write a predicate `inverttri/1`, the predicate `inverttri(N)` yields an "inverted triangular pattern" as in the following test cases:

- `?- inverttri(4).` returns

```
* * * *
* * *
* *
*
```

- `?- inverttri(N).` returns false for any integer `N` $<$ `1`.

4.2 Normal Triangular Pattern

(This subproblem is worth **10 points**.)

Write a predicate `startri/1`, the predicate `startri(N)` returns a "triangular pattern" as in the following test cases:

- `?- startri(4).` returns

```
*
* *
* * *
* * * *
```

- `?- startri(N).` returns false for any integer `N` $<$ `1`.

5 *Genap* and *Ganjil*

Remark 5 This problem is worth **20 points**.

Bagus is an informatics student at Telkom University. One day his sister—who is currently a third grader—ask him for assistance in a mathematics homework. The objective of this homework is to find out the parity of a particular integer or an elementary arithmetic expression involving integers. Some of the problems in this homework (in Bahasa Indonesia) are as follows:

12 genap (Y/T)	$2 \times (3 + 4)$ ganjil (Y/T)
28 ganjil (Y/T)	$1 - 3 \times 5$ ganjil (Y/T)
17 ganjil (Y/T)	$1 + 2 \times 4$ genap (Y/T)
14 genap (Y/T)	$(1 - 3) \times 5$ genap (Y/T)
$2 \times 3 + 4$ genap (Y/T)	$(1 + 2) \times 4$ ganjil (Y/T)

In Bahasa Indonesia, *genap* means even and *ganjil* means odd. Therefore, an integer N is *genap* if N is divisible by 2 and N is *ganjil* if N is not divisible by 2. Bagus's sister needs to circle *Y* if the statement is true and *T* otherwise.

Since Bagus is busy, instead of helping her sister in a one-on-one discussion, he makes a Prolog program to determine the parity of an integer or integer expression. The input of this program is an integer or arithmetic expression involving integers, followed by a blank space, and finally ended by an operator `genap` or `ganjil` and a period. The program outputs **true** if the statement is true and **false** otherwise. The arithmetic operations are limited to addition, subtraction, and multiplication. Some test cases are:

- `?- 12 genap.` returns **true**.
- `?- 12 ganjil.` returns **false**.
- `?- 2019 genap.` returns **false**.
- `?- 2019 ganjil.` returns **true**.
- `?- -2 * -3 + -4 genap.` returns **true**.
- `?- -2 * -3 + -4 ganjil.` returns **false**.
- `?- 1 + 2 * 4 genap.` returns **false**.
- `?- 1 + 2 * 4 ganjil.` returns **true**.
- `?- (1 + 2) * 4 genap.` returns **true**.
- `?- (1 + 2) * 4 ganjil.` returns **false**.

Hint 2 In Prolog documentation, `*` is an infix operator of type xfx and its precedence is 400, whereas both `+` and `-` can be treated as infix (yfx) or prefix (fx) operators, both of their precedences are 500. To solve the problem, we observe that the predicate `genap` and `ganjil` can be treated as a postfix operators.