Pra-Praktikum IF1221 Logika Komputasional

Dipersiapkan oleh Tim Asisten Lab Intelegensi Buatan

PETUNJUK PRA-PRAKTIKUM:

1. Pra-Praktikum IF1221 - Logika Komputasional adalah kegiatan yang bersifat **mandiri**.
2. Dilarang keras menggunakan tools dan AI seperti ChatGPT, Claude, Gemini, Bard, Llama, dan sebagainya dalam pengerjaan tugas. **Semua bentuk kecurangan akan ditindaklanjuti sesuai dengan sanksi akademik yang ada**.
3. Praktikum dikerjakan dengan menggunakan **GNU Prolog**. **Tidak diperbolehkan** menggunakan **SWI Prolog**.
4. Format file jawaban yang dikumpulkan adalah **PraPrak_[NIM].pl** untuk file prolog, dan **PraPrak_[NIM].txt** untuk query beserta hasilnya.
5. Sebelum menjawab tiap butir soal, tandai terlebih dahulu untuk setiap poin.

`/*Untuk File .pl*/`
 	
`/* Bagian <X> */`
`/* Deklarasi Fakta */`
`<fakta>`
 	
`/* Deklarasi Rules */`
`predikat1(X):- predikat2(X)`
	`/*Untuk File .txt*/`
	
	`Bagian <X>`
	
	`Query: <query>`
	`<hasil query>`

6. Apabila terdapat pertanyaan lebih lanjut, harap ditanyakan melalui sheets QnA berikut**.** Pertanyaan yang diajukan secara personal ke asisten **tidak akan dijawab** untuk menghindari perbedaan informasi yang didapatkan oleh peserta praktikum.
7. Semua deliverable files jawaban praktikum **dikompres** ke dalam arsip dengan ekstensi **.zip**, lalu dikumpulkan melalui **Edunex.** Format penamaan file arsip praktikum adalah **PraPrak_ [NIM].zip**.
8. Pastikan file arsip praktikum sudah di **SAVE dan SUBMIT** melalui edunex.
9. Apabila terdapat kendala dalam pengumpulan di Edunex, silakan ***upload*** file arsip praktikum melalui *link* ***Google Form*** **berikut**, **dengan mengabari asisten terlebih dahulu**.
10. Disarankan untuk mengerjakan semua bagian dengan struktur yang rapi dan jelas, karena selain memudahkan penilaian oleh asisten, juga akan sangat membantu dalam proses *debugging*.
11. ***Deadline*** pengumpulan adalah **29 November 2024 pukul 12.21**

BAGIAN I: Fakta, *Rule*, dan *Query*

Gunakan gambar pohon keluarga ini untuk menjawab bagian ini.

1. (10 Poin) Fakta

Buatlah fakta-fakta dari pohon keluarga di atas dengan menggunakan **HANYA aturan fakta di bawah ini**. Tulislah dalam bahasa pemrograman prolog lalu simpan dalam file .pl sesuai aturan.

a. pria(X) : X adalah pria
b. wanita(X) : X adalah wanita
c. usia(X,Y) : X berusia Y
d. menikah(X,Y) : X menikah dengan Y
e. anak(X,Y) : X adalah anak Y

2. (10 Poin) *Rule*

Buatlah *rule*/aturan di bawah ini **TANPA membuat *rule*/fakta tambahan**. Tulislah dalam bahasa pemrograman prolog lalu simpan dalam file .pl sesuai aturan! Diperbolehkan menggunakan *rule* yang sudah didefinisikan butir soal lain:c

a. saudara(X,Y) : X adalah saudara kandung maupun tiri dari Y
b. saudaratiri(X,Y) : X adalah saudara tiri dari Y
c. kakak(X,Y) : X adalah kakak dari Y (kakak kandung maupun tiri)
d. keponakan(X,Y) : X adalah keponakan dari Y
e. mertua(X,Y) : X adalah mertua dari Y
f. nenek(X,Y) : X adalah nenek dari Y
g. keturunan(X,Y) : X adalah keturunan dari Y (anak, cucu, dan seterusnya)
h. lajang(X) : X adalah orang yang tidak menikah
i. anakbungsu(X) : X adalah anak paling muda
j. anaksulung(X) : X adalah anak paling tua
k. yatimpiatu(X) : X adalah orang yang orang tuanya tidak terdefinisi

Catatan: Pastikan pemanggilan *rule* akan memberikan semua solusi jawaban, contoh:

```
| ?- lajang(X).

X = aqua ? ;

X = eve ?

yes
```

3. (10 Poin) Query

Implementasi kalimat di bawah ini ke bentuk query prolog, kemudian tulis query dan hasilnya dalam file .txt sesuai aturan! **Dilarang membuat *rule* tambahan selain dari soal**. Kerjakan soal menggunakan fakta, logika, dan *rule* dari 1.1 atau 1.2. Sama dengan 1.2, pastikan pemanggilan *query* memberikan **seluruh solusi jawaban**.

a. Suami dari aihoshino
b. Kakek buyut dan nenek buyut dari eve
c. Paman dari orang tua francesco
d. Orang yang memiliki lebih dari satu istri
e. Saudara tiri tertua dari shelby
f. Orang tua yang memiliki tepat satu anak
g. Pria yang memiliki mertua berusia lebih dari 99 tahun
h. Orang yatim piatu dengan usia termuda
i. Orang diatas 60 tahun yang statusnya menikah dengan orang yang memiliki saudara lebih dari 1 orang
j. Seluruh keturunan qika yang memiliki tepat 2 keponakan

BAGIAN II: Rekursivitas

Buatlah *rule* untuk solusi permasalahan berikut **menggunakan pendekatan rekursif**. Jika **tidak** menggunakan rekursivitas maka akan terdapat **pengurangan nilai** meskipun jawaban yang diberikan benar. Pada bagian ini Anda **diperbolehkan membuat *rule* lain** untuk membantu pengerjaan.

1. (6 Poin) Exponent

A (Nilai A > 0) merupakan basis perpangkatan, sedangkan B merupakan pangkat dari A (Nilai B ≥ 0). exponent(A, B, X) akan mengeluarkan hasil perpangkatan A^B yaitu X. **Solusi permasalahan wajib memanfaatkan rekursivitas, dilarang keras menggunakan operator perpangkatan (**), namun boleh memakai operator perkalian (*).** Berikut contoh *query* dan hasil eksekusi *query*.

exponent(A, B, X)

`| ?- exponent(2, 3, X).`
 	
`X = 8`
 	
`yes`

2. (6 Poin) Growth

Growth adalah fungsi yang menggambarkan pertumbuhan jumlah item dalam suatu kumpulan dengan pola yang bervariasi setiap tahunnya. Pola ini diawali dengan jumlah item awal I (dengan nilai I > 0). Pada setiap tahun ke-T (dengan nilai T ≥ 0), jumlah item akan bertambah atau berkurang tergantung pada pola yang ditentukan oleh dua parameter (G dan H) dengan aturan sebagai berikut.

- Jika T adalah bilangan prima, jumlah item bertambah sebesar G.
- Jika T bukan bilangan prima, jumlah item berkurang sebesar H (dengan nilai H < I).

Fungsi growth(I, G, H, T, X) akan mengembalikan jumlah item X pada tahun ke-T sesuai mekanisme yang telah dijelaskan. Dalam pengerjaan persoalan ini, **diperbolehkan untuk membuat fungsi antara**.

growth(I, G, H, T, X)

`| ?- growth(100, 10, 5, 0, X).`
 	
`X = 100 ?`
 	
`yes`
`| ?- growth(100, 10, 5, 1, X).`
 	
`X = 95 ?`
 	
`yes`
`| ?- growth(100, 10, 5, 3, X).`
 	
`X = 115 ?`
 	
`yes`
`| ?- growth(100, 15, 15, 3, X).`
 	
`X = 115 ?`
 	
`yes`

3. (6 Poin) Si Imut Anak Nakal

Seorang pencuri bernama Si Imut suka mencuri buah dari kebun ke kebun. Suatu hari, setelah ia mencuri dari kebun jambu Mas Faiz, ia pun menghampiri kebun mangga Mang Fuad untuk mengambil beberapa mangga. Pohon di kebun mangga Mang Fuad sudah diberi nomor, dari 1 sampai N. Si Imut terbiasa mengambil buah secara berurutan dari nomor yang dipilihnya hingga akhir. Contohnya jika ia memilih pohon nomor 5 dan ada 10 pohon, maka Si Imut hanya akan mengambil buah dari pohon nomor 5 sampai 10. Namun, tidak semua pohon berbuah, hanyalah pohon-pohon tertentu saja. Mang Fuad tahu kelakuan Si Imut ini, karena merasa geram, Mang Fuad akhirnya memasang jebakan di beberapa pohon berupa zat beracun yang menyebabkan buah yang saat ini dibawa Si Imut jadi membusuk. Setiap ke kebun Mang Fuad, Si Imut juga selalu membawa buah hasil curiannya dari kebun Mas Faiz. Keadaan yang dihadapi Si Imut yang cerdik nan malang ini adalah:

- Jika pohon yang dikunjungi adalah pohon yang nomornya habis dibagi 3, maka ia akan mendapatkan tambahan 2 buah.
- Jika pohon yang dikunjungi adalah pohon yang nomornya habis dibagi 4, maka buah yang ia bawa akan berkurang 5 buah.
- Jika pohon yang dikunjungi adalah pohon yang nomornya habis dibagi 5, maka ia akan mendapatkan tambahan 3 buah.
- Jika pohon yang dikunjungi adalah pohon yang nomornya adalah bilangan prima, maka buah yang ia bawa akan berkurang 10 buah.
- Jika buah mencapai 0 atau kurang sebelum Si Imut berhasil sampai ke pohon terakhir, Si Imut langsung pulang sambil menangis.

Implementasikan fungsi harvestFruits(N, Fruits, TreeNumber, FinalFruits) yang menggunakan rekursivitas untuk menghitung total buah yang didapatkan Si Imut selama aksi pencuriannya. Si Imut mulai dari nomor pohon yang dipilihnya (TreeNumber) dan dapat menjelajahi semua pohon hingga (N). Dalam pengerjaan persoalan ini, **diperbolehkan untuk membuat fungsi antara**.

harvestFruits(N, Fruits, TreeNumber, FinalFruits)

```
| ?- harvestFruits(20, 100, 1, FinalResults).
FinalResults = 19 ?
yes

| ?- harvestFruits(50, 500, 1, FinalResults).
FinalResults = 352 ?
yes

| ?- harvestFruits(50, 100, 1, FinalResults).
Si Imut pulang sambil menangis :(
FinalResults = 0 ?
yes
```

4. (6 Poin) KPK

X adalah kelipatan persekutuan terkecil(KPK) dari A dan B. Berikut ini adalah contoh query dan hasil query untuk rule berikut. Dalam pengerjaan persoalan ini, **diperbolehkan untuk membuat fungsi antara.** (Hint: gunakan hubungan antara FPB dan KPK)

KPK

```
| ?- kpk(23, 12, X).

X = 276 ?

yes
```

5. (6 Poin) Factorial

N (Nilai N ≥ 0) adalah bilangan yang akan dihitung faktorialnya. factorial(N, X) akan mengeluarkan hasil faktorial dari N, yaitu X. Solusi permasalahan wajib memanfaatkan rekursivitas. Berikut contoh query dan hasil eksekusi query :

factorial(N, X)

```
| ?- factorial(6, X).

X = 720 ?

yes
| ?- factorial(3, X).

X = 6 ?

yes
| ?- factorial(0, X).

X = 1 ?

yes
```

6. (6 Poin) Make Pattern

makePattern(N) akan menuliskan sebuah persegi angka sebesar N satuan dengan setiap angkanya menyatakan kedalaman persegi. Berikut ini adalah contoh *query* dan hasil *query* untuk *rule* berikut. Dalam pengerjaan persoalan ini, **diperbolehkan untuk membuat fungsi antara**. (Tips: gunakan fungsi write())

makePattern(N)

```
| ?- makePattern(5).

1 1 1 1 1
1 2 2 2 1
1 2 3 2 1
1 2 2 2 1
1 1 1 1 1
```

```
| ?- makePattern(1).

1
```

```
| ?- makePattern(6).

1 1 1 1 1 1
1 2 2 2 2 1
1 2 3 3 2 1
1 2 3 3 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

BAGIAN III: *List*

Buatlah *rule* untuk solusi permasalahan *list* berikut. Kerjakan bagian ini menggunakan **operator *pipe* (|)**. **Penggunaan *rule list processing*** yang disediakan oleh gprolog akan **membuat nilai anda menjadi 0**. Pembuatan ***rule* tambahan diperbolehkan** untuk mempermudah pekerjaan. Anda juga diperbolehkan untuk tidak mengikuti format query yang diberikan pada contoh query dan hasilnya selama di hasil pekerjaan anda **dijelaskan di komentar terkait struktur dan metode pemanggilan querynya! Asumsikan bahwa paling tidak terdapat 1 buah elemen di dalam *list* untuk semua pertanyaan di bawah ini!**

1. (10 Poin) *List Statistic*

Implementasikan operasi-operasi statistik dasar di bawah ini untuk membantu perhitungan statistik dari data yang dimiliki.

- **min**: mencari elemen dengan nilai minimum
- **max**: mencari elemen dengan nilai maksimum
- **range**: mencari selisih antara elemen terbesar dan elemen terkecil
- **count**: mencari jumlah elemen pada list
- **sum**: mencari jumlah total elemen pada list

Berikut ini adalah contoh *query* dan hasil *query* untuk *rule-rule* di atas

min(List, Min)

`| ?- min([5, -8, 4, 2, -1, 12], Min).`
 	
`Min = -8?`
 	
`yes`

max(List, Max)

`| ?- max([5, -8, 4, 2, -1, 12], Max).`
 	
`Max = 12?`
 	
`yes`

range(List, Range)
`| ?- range([5, -8, 4, 2, -1, 12], Range).`

```
Range = 20?

yes
```

count(List, Count)

```
| ?- count([5, -8, 4, 2, -1, 12], Count).

Count = 6?

yes
```

sum(List, Sum)

```
| ?- sum([5, -8, 4, 2, -1, 12], Sum).

Sum = 14?

yes
```

2. List Manipulation

a. (6 Poin) mergeSort

Merge Sort adalah algoritma *sorting* populer yang menggunakan strategi *divide and conquer*. Algoritma ini terbagi menjadi 3 bagian utama, yaitu *divide, conquer,* dan *merge.* Pada bagian *merge*, terdapat 2 list yang akan diambil nilainya satu persatu untuk kemudian disatukan ke dalam 1 list yang terurut. Berikut ilustrasinya.

ListA = [3, 7, 10]
ListB = [1, 8, 5]
Result = []

Step	ListA	ListB	Elemen terpilih	Result
1	3, 7, 10	1, 2, 5	1	1
2	3, 7, 10	2, 5	2	1, 2
3	3, 7, 10	5	3	1, 2, 3
4	7, 10	5	5	1, 2, 3, 5

5	7, 10		7	1, 2, 3, 5, 7
6	10		10	1, 2, 3, 5, 7, 10

Buatlah implementasi dari bagian *merge* pada algoritma *merge sort*.

mergeSort(ListA, ListB, Result)

`| ?- mergeSort([-5, 8, 14], [-2, -1, 12], Result).`
 	
`Result = [-5,-2,-1,8,12,14] ?`
 	
`yes`

b. (6 Poin) filterArray

Filter Array adalah fungsi yang digunakan untuk memfilter elemen-elemen dari sebuah array berdasarkan dua kriteria:

1. Elemen harus lebih besar dari Element1.
2. Elemen harus kelipatan dari Element2.

Fungsi ini akan menelusuri elemen-elemen dalam list satu per satu dan hanya menyertakan elemen-elemen yang memenuhi kedua syarat tersebut ke dalam Result.

filterArray(List, Element1, Element2, Result)

`| ?- filterArray([9, 14, 21, 4, 18, 12], 10, 3, Result).`
 	
`Result = [21, 18, 12] ?`
 	
`yes`

c. (6 Poin) reverse

Reverse adalah proses membalik urutan elemen sehingga elemen yang berada di posisi paling akhir menjadi yang pertama dan sebaliknya.

reverse (List, Result)

`| ?- reverse([29, 8, 31, 12], Reverse).`
 	
`Reverse = [12, 31, 8, 29] ?`
 	
`yes`

d. (6 Poin) cekPalindrom

Deskripsi

Buat sebuah fungsi cekPalindrom/1 dalam Prolog yang berfungsi untuk memeriksa apakah sebuah bilangan bulat adalah **palindrom**. Sebuah bilangan disebut **palindrom** jika angka-angka dalam bilangan tersebut sama saat dibaca dari depan maupun dari belakang. Sebagai contoh, list [1, 2, 1] adalah palindrom karena tetap sama jika dibaca terbalik, sedangkan [1, 2, 3] bukan **palindrom**.

Batasan

- Anda **tidak boleh menggunakan** fungsi bawaan Prolog seperti reverse/2 untuk membalik list.
- **Diperbolehkan menggunakan fungsi antara.**

cekPalindrom(List)

`?- cekPalindrom([1,2,2,1]).`
 	
`true ?`
 	
`atau`
 	
`?- cekPalindrom([1,2,2,1]).`
 	
`yes`

`?- cekPalindrom([0,1]).`
 	
`no`

e. (6 Poin) rotate

rotate(List, N, Result) adalah fungsi untuk memutar elemen dalam list sebanyak N kali ke arah kiri. Jika N lebih besar dari panjang list, jumlah rotasi akan dihitung dengan cara N mod Length, di mana Length adalah panjang list. Solusi permasalahan dilarang menggunakan fungsi bawaan seperti length/2 atau append/2. Perhatikan juga constraint berikut :

1. List kosong ([]) harus ditangani dengan mengembalikan list kosong.
2. Jika N < 0, rotasi ke arah kanan dilakukan dengan mengonversi N ke rotasi ke kiri yang ekuivalen.
3. Jika N > Panjang List, silahkan normalisasi jumlah rotasi.

Berikut contoh query dan hasil eksekusi query

rotate(List, N, Result)

```
| ?- rotate([], 3, Result).

Result = [] ?

yes
| ?- rotate([1, 2, 3, 4], 2, Result).

Result = [3,4,1,2] ?

yes
| ?- rotate([1, 2, 3, 4], 6, Result).

Result = [3,4,1,2] ?

yes

| ?- rotate([1, 2, 3, 4, 5], -2, Result).

Result = [4,5,1,2,3] ?

yes
```

f. (6 Poin) Mapping

Pak Amba adalah salah satu dosen di Universitas Garut (UNIGA). Setiap di akhir semester, Pak Amba selalu memberikan nilai akhir kepada mahasiswanya. Namun, Pak Amba tidak bisa memberikan nilai dalam bentuk angka, melainkan harus dalam bentuk huruf. Berikut adalah konversi nilai angka ke huruf:

- Jika nilai >= 80, maka nilai hurufnya adalah A
- Jika nilai >= 70, maka nilai hurufnya adalah B
- Jika nilai >= 60, maka nilai hurufnya adalah C
- Jika nilai >= 50, maka nilai hurufnya adalah D
- Jika nilai < 50, maka nilai hurufnya adalah E

Bantulah Pak Amba dengan membuat sebuah program yang dapat mengkonversi array of nilai dalam bentuk angka ke array of nilai dalam bentuk huruf/indeks. Program tersebut akan menerima input berupa nama mahasiswa dan nilai dari mahasiswa tersebut. Program akan mengkonversi nilai-nilai tersebut ke dalam bentuk huruf dan menghitung nilai rata-rata mahasiswa tersebut. Program akan mengeluarkan output berupa nama mahasiswa, nilai-nilai mahasiswa dalam bentuk huruf, nilai rata-rata, dan status kelulusan mahasiswa tersebut. Jika nilai rata-rata >= 80, maka status kelulusan adalah "Pass", jika tidak, maka status kelulusan adalah "Fail". **Dilarang menggunakan fungsi bawaan apapun.**

Hint: Untuk konversi nilai ke indeks, buatlah fungsi *mapping* yang akan mengonversi elemen di tiap *array of nilai* ke dalam bentuk indeks, penggunaan percabangan diperbolehkan, buatlah fungsi length() untuk mencari panjang array of nilai dan fungsi average() dengan memanfaatkan fungsi SUM yang telah dibuat sebelumnya untuk mencari rata-rata nilai akhir.

prosesMahasiswa(Name, Grades, Result)

Contoh eksekusi 1:

```
| ?- prosesMahasiswa('Rusdi', [5, 0, 82], Result).

Result = ['Rusdi',['E','E','A'],29.0,'Fail']

yes
```

Contoh eksekusi 2:

```
| ?- prosesMahasiswa('Rusdi', [5, 0, 82, 90, 88, 23, 81,
78, 65, 90 ,100, 100], Result).
```

```
Result =
['Rusdi',['E','E','A','A','A','E','A','B','C','A','A','A'
],66.833333333333329,'Fail']

yes
```

Contoh eksekusi 3:

```
| ?- prosesMahasiswa('Fuad', [90, 88, 60, 81, 78, 65, 90
,100, 100], Result).

Result =
['Fuad',['A','A','C','A','B','C','A','A','A'],83.55555555
5555557,'Pass']

yes
```

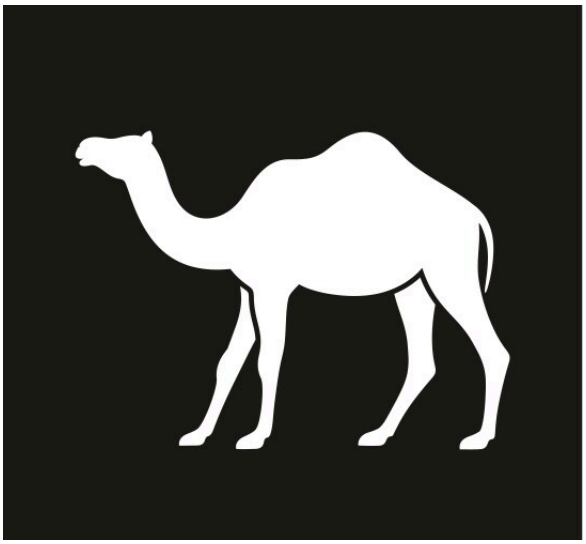
Contoh eksekusi 4:

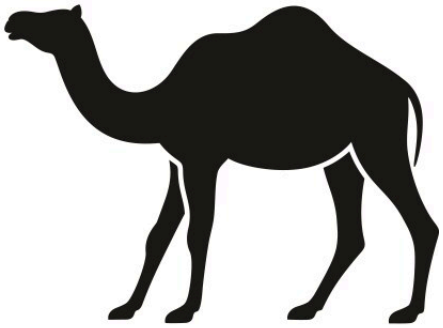
```
| ?- prosesMahasiswa('Fuad', [0], Result).

Result = ['Fuad',['E'],0.0,'Fail']

yes
```

(10 poin) BONUS

Bonus ini disarankan untuk dikerjakan untuk mempermudah peserta kuliah dalam pengerjaan tugas besar.

Petualangan Mencari Koin





Kamu terbangun di padang pasir dengan hati yang berat. Unta putih dan unta hitam, sahabat setiamu selama bertahun-tahun, kini berada dalam kendali para pedagang karavan. Mereka bersedia mengembalikan untamu jika kamu dapat membuktikan kemampuanmu dalam menjelajahi reruntuhan kuno gurun ini untuk mengumpulkan koin sebanyak mungkin. Akan tetapi, perjalanan ini tidak mudah. Para pencuri makam telah meninggalkan jebakan di setiap sudut reruntuhan. Setiap langkah yang kamu ambil bisa menjadi berbahaya, tergantung pada keberuntunganmu. Kamu ingin melakukan simulasi sebelum mulai melakukan penjelajahan, oleh karena itu buatlah sebuah simulasi permainan sederhana dengan ketentuan sebagai berikut :

Simulasi Perintah Permainan

Pastikan mendukung perintah-perintah berikut :

1. **start**
   Memulai permainan. Koin awal diatur ke 0. Menampilkan pesan pengantar tentang tujuan permainan.

2. **move(Direction)**
   Menggerakkan pemain ke arah tertentu (north, south, east, atau west). Setiap langkah memiliki dua kemungkinan:
   - Jalur Aman: Koin bertambah 10.
   - Terkena Jebakan: Koin berkurang 10.

   Skor dapat menjadi negatif.
3. **display_status**
   Menampilkan informasi koin terkini.
4. **exit**
   Untuk keluar dari permainan. Jika permainan belum dimulai, tampilkan pesan kalau permainan belum dimulai.

Catatan Permainan

1. Jebakan (Trap)
   Setiap langkah memiliki peluang **25% terkena jebakan**. Jika terkena jebakan, koin berkurang 10. Jika jalan aman, koin bertambah 10.
2. *Randomization*
   Program menggunakan fungsi bawaan random untuk menentukan hasil langkah (jalur aman atau jebakan).

*) Perintah adalah fungsi yang tidak memiliki parameter apapun.
*) Pesan-pesan yang ditulis harus sama persis dengan contoh.

Contoh Jalannya Permainan

Sebelum Game Dimulai

```
?- move(north).

no

?- display_status.

no

?- exit.
Permainan belum dimulai. Gunakan "start" untuk memulai.

no
```

Memulai Game

```
| ?- start.
Unta putih dan unta hitammu diambil oleh pedagang
karavan. Kumpulkan koin sebanyak-banyaknya !!!

yes
| ?- start.
Permainan sudah dimulai. Gunakan "exit" untuk keluar dan
memulai ulang.

yes
```

Melihat Status

```
| ?- display_status.
Koin Saat Ini: 0
```

Bergerak Terkena Jebakan

```
| ?- move(east).
Kamu bergerak ke arah east.
Oh tidak! Kamu terkena jebakan!
Koin Saat Ini: -10

yes
```

Bergerak tanpa Terkena Jebakan

```
| ?- move(north).
Kamu bergerak ke arah north.
Jalannya aman.
Kamu mendapatkan 10 poin.
Koin Saat Ini: 0

yes
```

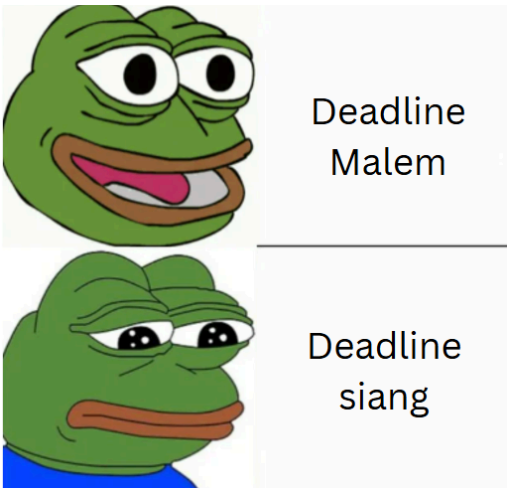
Keluar dari Permainan

```
| ?- exit.
Terima kasih telah memainkan simulasi ini! Sampai jumpa
lagi.

yes
```

Referensi

GNU Prolog Manual : www.gprolog.org/manual/gprolog.html

Catatan Asisten



Semangat mengprakprakk…


SEMANGAT
siapa
suruh
MASUK ITB?


kamu kan
kamu kan
anak STEI
masa gini aja
gak bisa?


SUDAH SAATNYA

mangat gess :)))