

Spesifikasi Tugas Besar

IF1221 Logika Komputasional

Tahun 2024/2025

Camel POP!

Dipersiapkan Oleh Tim Asisten Lab Intelelegensi Buatan 2022

Waktu Mulai: Jumat, 29 November 2024

Waktu Selesai: Jumat, 20 Desember 2024 pukul 12.21

Revisi 1.0 (05/12/2024 20.10):

Kondisi kartu setelah investasi, urutan unta bertumpuk menang di petak *finish*

Revisi 1.1 (06/12/2024 03.54):

Display map

Revisi 1.2 (06/12/2024 14.30):

Jumlah trap, output poin pada endgame, referensi, batas petak unta putih dan hytam, god's hand, tukar unta, sembunyikan unta

Topik

Membuat sebuah *board game* dengan menggunakan bahasa pemrograman deklaratif Prolog (GNU Prolog).



sumber : <https://www.gameliikeamother.com/camel-up>

Implementasi tugas besar **harus** mengandung materi:

1. Rekurens
2. *List*
3. *Cut*
4. *Fail*
5. *Loop*

Tujuan

Tujuan dari tugas besar ini adalah mengkombinasikan berbagai keterampilan dan teknik yang telah dipelajari dalam perkuliahan Logika Komputasional IF2121, pra-praktikum, dan eksplorasi mandiri mengenai Logika Komputasional dan Prolog.

Timeline

No	Waktu	Kegiatan
1	Jumat, 29 November 2024	Pembahasan Pra-Praktikum, Rilis Tugas Besar IF2121.
2	Rabu, 4 Desember 2024	Batas pengisian daftar kelompok.

3	Jumat, 6 Desember 2024 pukul 12.21	Milestone 1: Asistensi Tugas Besar, penggerjaan, dan pengumpulan <i>progress</i> pertama tugas besar.
4	Jumat, 13 Desember 2024 pukul 12.21	Milestone 2: Asistensi Tugas Besar, penggerjaan, dan pengumpulan <i>progress</i> kedua dari tugas besar.
5	Jumat, 20 Desember 2024 pukul 12.21	Milestone 3: Pengumpulan <i>final deliverables</i> Tugas Besar Logika Komputasional.
6	[menyusul]	Batas pengisian jadwal demo tugas besar.
7	[menyusul]	Pelaksanaan demo tugas besar.

Alur Permainan

1. Permainan diinisiasi dengan [start game](#). Command ini akan melakukan *setup* berupa pengacakan urutan pemain serta inisiasi kartu, poin, dan *trap* tiap pemain.
2. Pada tiap giliran pemain, pemain dapat melakukan:
 - pilih maksimal 1 aksi di antara: [investasi](#), [jalankan unta](#), [pasang trap](#) (juga [sembunyikan unta](#), [restart unta](#), [tukar unta](#), atau [God's Hand](#) jika mengerjakan bonus);
 - [display map](#) dengan jumlah tidak terbatas;
 - [cek info](#) dengan jumlah tidak terbatas.Pemain dapat mengakhiri gilirannya dengan menjalankan [end turn](#).
3. Permainan akan berakhir ketika terdapat unta yang melewati garis *finish* (angka dadu untuk melewati garis *finish* **tidak harus tepat** dengan jumlah langkah yang dibutuhkan untuk sampai ke garis *finish*). Program secara otomatis akan menjalankan [end game](#).

Spesifikasi

A. Start Game (*Initiating*)

Tahap ini berisi seputar *setup* awal permainan, hal ini meliputi pembuatan *map*, pemain, dan inisialisasi dadu.

- **Pemain**

Permainan dapat dimainkan oleh 2-4 pemain dengan urutan pemain ditentukan secara acak. Setelah memasukkan jumlah pemain yang diinginkan, pemain dapat memasukkan nama pemain dan nama tersebut harus unik.

- **Kartu**

Dalam permainan, setiap pemain memiliki 4 kartu yang memiliki warna merah, kuning, hijau, dan biru. Setiap warna pada kartu ini melambangkan unta yang dipilih. Pemain harus menentukan urutan unta yang nanti akan dijadikan acuan dalam penentuan poin.

- **Unta**

Dalam permainan, terdapat 5 unta yang memiliki warna merah, kuning, hijau, biru, dan putih (6 unta dengan tambahan unta berwarna hitam jika mengerjakan bonus).

- **Dadu**

Dalam permainan, terdapat 5 dadu yang memiliki warna merah, kuning, hijau, biru, dan putih. Masing-masing dadu memiliki angka dari 1 sampai 6 (khusus dadu putih akan menjadi dadu hitam dengan angka 1, 3, 5 dan dadu putih dengan angka 2, 4, 6 jika mengerjakan bonus).

- **Poin**

Kemenangan pemain ditentukan oleh poin. Pemenang dengan poin terbanyak pada akhir permainan akan menjadi pemenang. Pada awal permainan, setiap pemain memiliki 30 poin.

- **Trap**

Trap merupakan jebakan yang dapat menambah poin seorang pemain. Untuk penjelasan lebih lengkap dapat dibaca pada bagian [pasang trap](#). Pada awal permainan, setiap pemain memiliki 1 trap.

Contoh Program

```
| ?- startGame.  
  
Masukkan jumlah pemain: 5.  
Mohon masukkan angka antara 2 - 4.  
Masukkan jumlah pemain: 3.  
  
Masukkan nama pemain 1: Eve.  
Masukkan nama pemain 2: Cesco.  
Masukkan nama pemain 3: Qika.  
  
Urutan pemain: Qika - Eve - Cisco.  
  
Setiap pemain mendapatkan 4 kartu, 30 poin dan 1 trap.  
  
Kartu Qika: merah, kuning, hijau, biru.  
Kartu Eve: merah, kuning, hijau, biru.  
Kartu Cesco: merah, kuning, hijau, biru.  
  
Poin Qika: 30.  
Poin Eve: 30.  
Poin Cesco: 30.  
  
Trap Qika: 1.  
Trap Eve: 1.  
Trap Cesco: 1.  
  
(69 ms) yes
```

B. Turn

Setelah tahap [initiating](#) dilakukan, selanjutnya setiap pemain akan bermain secara bergiliran. Tiap giliran pemain, pemain dapat memilih **tepat satu action**, dengan opsi: [investasi](#), [jalankan unta](#), [pasang trap](#) (juga [sembunyikan unta](#), [restart unta](#), [tukar unta](#), atau [God's Hand](#) jika mengerjakan **bonus**). Selain itu, dalam tiap giliran, pemain dapat meminta program untuk **menampilkan map** dan **menampilkan info** dengan jumlah **tidak terbatas**.

1. Map

Buatlah board 5x5 yang akan digunakan sebagai arena permainan dengan blok pada pojok kiri atas sebagai start dan finish unta.

Penjelasan Board 5x5

1. Setiap kotak mewakili petak tempat unta bisa berhenti.
2. Nama kotak mengikuti format abjad dari A sampai **O**.
3. Setiap kotak memiliki tanda kurung siku dan tanda kurung biasa.
4. Tanda kurung biasa ada di bawah kurung siku.
5. Kotak pertama (S/F) adalah titik awal dan akhir unta.
6. Unta bergerak sesuai urutan abjad.

Aturan Map:

1. Petak Kosong: Jika tidak ada unta di petak, kurung siku kosong.
2. Jika ada lebih dari satu unta di petak dan mereka saling bertumpuk, **gunakan tanda koma dan spasi untuk memisahkan unta pada maps.**
3. Pada kasus bertumpuk, unta yang dituliskan lebih dahulu (di sebelah kiri) berada di tumpukan paling bawah, sedangkan unta yang dituliskan setelahnya berada di atasnya.
Contoh : [UM, UK, UB] artinya unta biru (UB) berada di atas unta kuning (UK), kemudian unta kuning (UK) berada di atas unta merah (UM).
4. Jika ada lebih dari satu unta di petak dan mereka tidak saling bertumpuk, **gunakan spasi memisahkan unta pada maps.**
5. Visualisasi ‘Trap’ pada maps dituliskan pada tanda kurung biasa. Untuk trap maju tambahkan tanda > dan untuk trap mundur tambahkan tanda <.
 - (Trap>) untuk trap maju
 - (<Trap) untuk trap mundur
6. Map tidak harus sama persis, tapi **wajib terlihat jelas, tidak terlalu kecil, dan memuat semua komponen yang ada.**
7. Ilustrasi:

S/F	A	B	C	D
O				E ▼
N				F
M				G
L	K	J	I	H

Keterangan Singkatan Unta:

- UM: Unta Merah
- UK: Unta Kuning
- UH: Unta Hijau
- UB: Unta Biru
- UP: Unta Putih
- UI: Unta Item

Kondisi Map pada awal permainan

Contoh Program				
<pre> ?- displayMap. +-----+-----+-----+-----+-----+ S/F [] [] [] [] [UB UK UH UM UP UI] () () () () +-----+-----+-----+-----+-----+ O [] [] [] [] [] () () () () +-----+-----+-----+-----+-----+ N [] [] [] [] [] () () () () +-----+-----+-----+-----+-----+ M [] [] [] [] [] () () () () +-----+-----+-----+-----+-----+ L [] [] [] [] [] () () () () +-----+-----+-----+-----+-----+ (31 ms) yes</pre>				

Kondisi :

- UB berada di petak F.
- UH berada di petak A.
- UM dan UK saling bertumpuk di petak C.
- UI ada di petak M.
- UP ada di petak L.
- Ada Trap di petak I.

Contoh Program					
<pre> ?- displayMap. +-----+ S/F [UB] A [] B [] C [UM, UK] D [] () () () () () +-----+ O [] E [] () () +-----+ N [] CAMEL UP! () +-----+ M [] F [] () () +-----+ L [] G [] () () +-----+ K [] J [] I [Trap>] H [] () () () () +-----+ yes</pre>					

2. Cek Info

Buatlah sebuah fungsi untuk menampilkan detail dari suatu pemain, seperti nama pemain, poin yang dimiliki, sisa kartu, dan sisa trap. Fungsi dapat dipanggil kapan saja, fungsi bisa saja digunakan pada diri sendiri.

Contoh Program	
<pre> ?- cek_info. Masukkan nama pemain: panji. Nama pemain: panji Poin: 3 Kartu: [merah,kuning,hijau,biru] Trap: 1 yes</pre>	<pre> ?- cek_info.</pre>

```
Masukkan nama pemain: eriq.  
Nama pemain: eriq  
Poin: 2  
Kartu: [merah,kuning,hijau,biru]  
Trap: 1  
  
yes
```

3. Investasi

Fitur Investasi memungkinkan pemain untuk memilih unta yang diharapkan dapat mendapatkan juara. Setiap pemain hanya dapat berinvestasi pada satu unta per giliran. Investasi memberikan poin tambahan di akhir permainan berdasarkan urutan *finish* unta yang diinvestasikan. Jika lebih dari satu pemain berinvestasi pada unta yang sama, poin dibagi sesuai dengan urutan investasi. Detail terkait poin perolehan investasi akan dijelaskan pada bagian [End Game](#). Berikut adalah ilustrasi cara kerja investasi:

Misalkan Beatrice berinvestasi pada unta merah:

1. Beatrice
2.
3.
4.

Kemudian pemain berikutnya, yaitu Morgana, memilih investasi juga pada unta merah, maka urutan investasi pada unta merah saat ini akan menjadi:

1. Beatrice
2. Morgana
3.
4.

Aturan Investasi:

1. Pemain hanya boleh melakukan investasi pada gilirannya.
2. Pemain tidak diperbolehkan melakukan investasi ulang pada unta yang sama.
3. Pemain hanya dapat memilih investasi pada unta dengan warna yang valid, yaitu merah, kuning, hijau, atau biru.
4. Setelah melakukan investasi, kartu dengan warna yang sama dengan warna unta yang diinvestasikan akan dihapus. Misalnya sebelum berinvestasi pada unta merah, kartu Morgana: merah, kuning; setelah berinvestasi pada unta merah, kartu Morgana: kuning.

Contoh Program

Kasus Morgana mencoba melakukan investasi

```
| ?- investasi.
```

Pilih warna unta untuk investasi (merah, kuning, hijau, biru): merah.
Investasi berhasil! Morgana telah berinvestasi pada unta merah.
Morgana masuk ke daftar investasi unta merah.
Papan investasi pada unta merah saat ini:

```
+-----+  
| INVESTASI UNTA |  
+-----+  
1. Beatrice  
2. Morgana
```

yes

Kasus Beatrice mencoba melakukan investasi pada unta yang sama

```
| ?- investasi.
```

Pilih warna unta untuk investasi (merah, kuning, hijau, biru): merah.
Gagal! Beatrice sudah berinvestasi pada unta merah sebelumnya.
Papan investasi pada unta merah saat ini:

```
+-----+  
| INVESTASI UNTA |  
+-----+  
1. Beatrice  
2. Morgana
```

yes

```
| ?- investasi.
```

Pilih warna unta untuk investasi (merah, kuning, hijau, biru): putih.
Gagal! putih bukan warna unta yang valid.
Silakan pilih warna yang valid (merah, kuning, hijau, biru).

yes

Pemain bisa menampilkan papan investasi untuk semua unta

| ?- papan_investasi.

Papan investasi pada unta merah:

```
+-----+  
| INVESTASI UNTA |  
+-----+  
1. Beatrice  
2. Morgana  
3. SCP 040 JP
```

Papan investasi pada unta kuning:

```
+-----+  
| INVESTASI UNTA |  
+-----+  
1. Alice  
2. Morgana
```

Papan investasi pada unta biru:

```
+-----+  
| INVESTASI UNTA |  
+-----+  
1. Beatrice
```

Papan investasi pada unta hijau:

```
+-----+  
| INVESTASI UNTA |  
+-----+  
Belum ada investasi
```

yes

4. Jalankan Unta

Jika pemain memilih jalankan unta, akan ada dua dadu yang dilempar, yaitu dadu warna dan dadu angka, Dadu warna terdiri dari warna merah, biru, hijau, kuning, dan putih, (**hitam(bonus)**). Sedangkan, dadu angka terdiri dari angka 1 sampai 6. **Dadu warna yang warnanya sudah keluar, tidak**

akan keluar lagi sampai seluruh warna dadu keluar. Misalnya, ketika warna merah keluar, maka ia baru bisa keluar lagi setelah biru, hijau, dan kuning keluar.

Saran: Cukup buat 1 fungsi saja untuk mengocok kedua jenis dadu, misal fungsi “kocokDadu()”.

Unta bergerak berdasarkan hasil kocokan dadu, unta yang bergerak adalah unta yang warnanya sesuai dengan hasil kocokan **dadu warna**, dan unta tersebut akan bergerak sebanyak angka yang muncul pada pengocokan **dadu angka**. Misalnya, setelah pengocokan, didapatkan dadu warna merah dan dadu angka 3, maka unta yang akan bergerak adalah unta berwarna merah sebanyak 3 langkah **searah jarum jam**.

Jika di petak tujuan sudah ada unta berwarna lain, maka unta yang bergerak akan **berada di atas unta yang sudah ada di petak itu**. Ketika unta yang saling bertumpukan bergerak, maka unta tersebut yang akan jalan akan membawa jalan unta yang berada di atasnya namun tidak dengan yang ada di bawahnya.

Untuk memudahkan, berikut ilustrasinya:

1. Diasumsikan unta merah masih berada di garis *start*. Misalkan, dadu dikocok dan didapatkan dadu warna merah dan dadu angka 3, maka unta berwarna merah akan bergerak menuju ke petak 3.

Unta Merah

2. Kemudian dilakukan pengacakan lagi dan didapatkan dadu warna merah lagi dengan dadu warna 4, unta merah akan bergerak menuju petak 7. Sedangkan di petak 7 sudah ada unta hijau, maka, unta merah akan berada di atas unta hijau.



3. Sesudah itu dadu dikocok kembali dan didapatkan dadu warna hijau dengan dadu angka 2. Maka unta hijau akan bergerak menuju petak 9 dengan membawa unta merah di atasnya. Di petak 9, ternyata ada unta biru. Maka, unta hijau dan unta merah akan berada di atas unta biru.



4. Kemudian, dari pengacakan selanjutnya, didapatkan dadu warna hijau dan dadu angka 1. Karena yang berada di atas unta hijau hanyalah unta merah dan unta biru ada di bawahnya. Maka, unta hijau akan bergerak menuju petak 10 hanya dengan membawa unta merah. Di petak 11, akan hanya ada unta merah dan unta hijau.



Setiap kali pemain memilih untuk menjalankan unta, maka, pemain tersebut akan mendapatkan **10 poin** apapun yang terjadi terlepas unta yang berjalan adalah unta yang ia investasikan atau bukan.

Khusus untuk unta putih (dan **hytam** jika mengerjakan bonus), jika sudah sampai di petak A dan tidak ada unta yang berada di bawahnya, maka unta tersebut tidak akan bisa bergerak lagi. Jika unta putih (dan **hytam**) berada di petak B dan mendapatkan angka kocokan dadu 2, maka unta tersebut akan terjebak di petak A, **tidak berjalan lebih**.

Note: Format output khususnya peta dan format tumpukan unta dibebaskan. Buatlah sekreatif mungkin.

Contoh Program	
<pre>?-jalankanUnta().\n\nDadu dikocok...\nDadu Angka: 5\nDadu Warna: merah\n\nUnta merah bergerak sebanyak 5 langkah...\n\nPeta:\n+-----+\n S/F [UB UH UP UI] A [UK] B [] C [] D [] \n () () () () () () () () () \n+-----+\n O [] () () () () () () () \n () () () () () () () () () \n+-----+\n N [] () () () () () () () \n () () () () () () () () () \n+-----+\n M [] () () () () () () () \n () () () () () () () () () \n+-----+\n L [] () () () () () () () \n () () () () () () () () () \n+-----+\n K [] () () () () () () () \n () () () () () () () () () \n+-----+\n J [] () () () () () () () \n () () () () () () () () () \n+-----+\n I [] () () () () () () () \n () () () () () () () () () \n+-----+\n H [] () () () () () () () \n () () () () () () () () () \n+-----+\n (16 ms) yes </pre>	

5. Pasang Trap

Pemain juga dapat memasang trap. Trap yang dipisah minimal berjarak 1 kotak dengan trap lainnya. Trap hanya dapat dipasang di petak dimana tidak terdapat unta di petak tersebut.

Trap Eve	Tidak dapat menaruh trap di sini	Dapat menaruh trap di sini	Tidak dapat menaruh trap di sini	Trap Cesco
----------	----------------------------------	----------------------------	----------------------------------	------------

Pemain yang memasang trap akan mendapatkan 10 poin jika ada unta yang mendarat di petak yang terpasang trap oleh pemain tersebut. Setelah itu, trap akan menghilang. Dengan kata lain tiap pemain hanya dapat memasang trap satu kali dan trap tersebut hanya dapat menjebak unta

sebanyak satu kali kemudian menghilang. Ketika unta mendarat di petak dengan trap maju, unta akan berpindah ke petak di depannya (misal dari petak C ke petak D), begitu pula sebaliknya.

Contoh Program

```
| ?- pasangTrap.  
  
Masukkan kode petak: A.  
Masukan jenis trap (maju/mundur): maju.  
  
Trap berhasil dipasang. Unta yang mendarat pada petak tersebut akan maju  
1 petak.  
Eve akan mendapatkan 10 poin ketika ada unta yang mendarat di petak  
tersebut.  
  
Sisa trap Eve: 0.  
  
(69 ms) yes  
  
| ?- pasangTrap.  
  
Masukkan kode petak: A.  
  
Trap gagal dipasang.  
Pastikan tidak ada 2 trap di petak yang bertetangga.  
  
Sisa trap Eve: 1.  
  
(69 ms) yes
```

6. End Turn

Pada setiap turn, pemain **hanya bisa memanggil 1 perintah**.
Namun, pemanggilan terhadap [map](#) dan [informasi](#) dapat dilakukan berkali-kali.

Contoh Program

```
| ?- next_turn.  
  
Sekarang giliran pemain B.
```

C. End Game

Permainan berakhir ketika terdapat satu unta yang melewati garis finish. Setelah permainan berakhir, hasil investasi akan dihitung dan ditambahkan ke poin setiap pemain. Perhitungan poin sebagai berikut.

Urutan Unta	Poin
1	50 / 30 / 20
2	10
3	5
4	0

Urutan Reward Investasi pada Unta yang Menang	Poin
1	50
2	30
3	20
4	20

Misalkan hasil akhir urutan unta sebagai berikut:

1. Merah
2. Kuning
3. Hijau
4. Biru

dan queue setiap investasi sebagai berikut.

Urutan	Merah	Kuning	Hijau	Biru
1	+50 poin (Eve)	+30 poin	+20 poin	+20 poin

2	+10 poin	+10 poin	+10 poin	+10 poin
3	+5 poin	+5 poin	+5 poin (Eve)	+5 poin
4	0 poin	0 poin	0 poin	0 poin

Maka Eve mendapat **tambahan poin (bukan poin akhir)** sebanyak $50 + 5 = 55$ poin.

Jika ada tumpukan unta pada petak *finish*, maka urutan unta yang menang terhitung dari atas. Misalnya pada petak *finish* terdapat tumpukan unta: merah (paling atas), kuning (tengah), hijau (paling bawah), maka urutan unta menang menjadi:

1. Unta merah
2. Unta kuning
3. Unta hijau
4. Unta biru (tidak sampai *finish*)

Contoh Program

Selamat, Eve menang dengan 55 poin!

(69 ms) yes

BONUS

1. Unta Aseli Ngawi

Buatlah unta berwarna **hytam** yang **bergerak secara mundur**, sama seperti unta putih, tujuan dari unta **hytam** ini adalah sebagai pengganggu dengan memundurkan posisi unta yang berada di atasnya. Unta **hytam** ini bergerak sesuai dengan dadu berwarna hitam putih, jika angka yang keluar adalah ganjil, gerakan unta **hytam** sesuai dengan angka yang keluar. Jika angka yang keluar adalah genap, gerakan unta putih sesuai dengan angka yang keluar.

Contoh Program

```
/* KASUS SEMUA UNTA BERADA DI ATAS UNTA HYTAM SEMUA (KECUALI UNTA PUTIH): */

| ?- kocok_dadu.
Dadu dikocok:
Dadu angka: 1
Dadu warna: abu-abu

Before movement:
Current positions:
Black camel: C
White camel: S/F
Red camel: C
Blue camel: C
Yellow camel: C
Green camel: C

Black camel moves!

After movement:
Current positions:
Black camel: B
White camel: S/F
Red camel: B
Blue camel: B
Yellow camel: C
Green camel: B

/* KASUS SEMUA UNTA BERADA DI ATAS UNTA HYTAM SEMUA (KECUALI UNTA PUTIH DAN HIJAU): */

| ?- kocok_dadu.
Dadu dikocok:
Dadu angka: 1
Dadu warna: abu-abu

Before movement:
Current positions:
Black camel: D
White camel: S/F
Red camel: D
Blue camel: D
Yellow camel: C
Green camel: D

Black camel moves!

After movement:
Current positions:
Black camel: C
White camel: S/F
Red camel: C
```

```
Blue camel: C  
Yellow camel: C  
Green camel: D
```

```
(16 ms) yes
```

Penjelasan:

- Baik unta **hytam** dan putih tidak bisa bergerak ketika sudah **next blocknya** adalah S/F block
- Unta **hytam** dan putih mulai dari S/F seperti unta lain pada umumnya, ingat unta **tidak bisa bertumpuk pada block S/F**
- Unta **hytam** dan putih mundur sebanyak angka dari dadu abu-abu yang keluar, ketika sudah di ujung (pada block A), unta **hytam** dan putih tidak bisa bergerak lagi.

2. Sembunyikan Unta

Buatlah action `sembunyikanUnta` yang menghilangkan 1 unta dengan warna acak (gunakan `kocokDadu()`), misal warna X, selama 1 turn. Unta akan dimunculkan kembali pada turn berikutnya. Misalkan ada setumpuk unta merah-kuning-hijau pada petak B dengan kondisi unta kuning disembunyikan, maka jika unta merah dan hijau bergerak bersamaan pada turn dimana unta kuning dihilangkan, maka ketika dimunculkan kembali, unta kuning akan tetap berada di petak B. Unta yang disembunyikan akan dimunculkan kembali **tepat sebelum giliran pemain berikutnya dimulai**.

Ingin: dadu dengan warna X juga menghilang 1 turn dari permainan. **Contoh implementasi menghilangkan dadu: (pilih salah satu saja)**

1. Mengulang ketika hasil acak dadu warna sama dengan warna X (aksi acak ulang tidak usah ditulis di layar)
2. Tidak dimungkinkan warna X muncul sama sekali pada proses acak dadu warna)

Contoh Program

```
?- sembunyikanUnta().  
  
Kocok dadu...  
Didapat warna: merah.  
Unta merah menghilang selama satu turn.
```

3. Restart Reversal Technique

Restart Reversal Technique adalah teknik strategis dengan risiko tinggi yang hanya dapat digunakan oleh pemain yang memenuhi syarat tertentu. Pertama, **pemain harus sudah berinvestasi pada unta yang berada di posisi paling bawah dalam sebuah tumpukan di petak tertentu**. Hal ini berarti investasi pemain harus secara khusus diarahkan pada unta yang menjadi fondasi tumpukan, bukan unta di posisi atas atau tengah. Kedua, **pemain harus menjadi investor pertama pada papan investasi untuk unta tersebut**, memastikan bahwa hak penggunaan teknik ini tidak tersedia bagi investor berikutnya dalam urutan investasi. Selain itu, **setiap pemain hanya dapat menggunakan Restart Reversal Technique satu kali selama permainan**.

Teknik ini memiliki peluang keberhasilan sebesar **50%**. Jika berhasil, unta yang berada di posisi paling atas tumpukan pada petak tersebut akan langsung dikirim kembali ke Start/Finish (S/F). Namun, jika gagal, pemain yang mengaktifkan teknik ini akan kehilangan 25 poin, tanpa ada perubahan pada posisi unta dalam papan permainan. Jika pemain memiliki lebih dari satu unta di posisi tumpukan paling bawah, Restart Reversal Technique tidak dapat digunakan.

Contoh Program

KASUS BERHASIL
Map sebelum

```
| ?- displayMap.
+-----+-----+-----+-----+-----+
| S/F |     A |     B |     C |     D |
| [ ] |     [ ] |     [ ] | [UM, UK, UB] |     [ ] |
| ( )|     ( )|     ( )| ( )           |     ( ) |
+-----+-----+-----+-----+-----+
| O  |           |           |           |     E |
| [ ]|           |           |           | [ ] |
| ( )|           |           |           | ( ) |
+-----+-----+-----+-----+-----+
| N  |           |           |           |     F |
| [ ]|           |           |           | [ ] |
| ( )|           |           |           | ( ) |
+-----+-----+-----+-----+-----+
| M  |           |           |           |     G |
| [ ]|           |           |           | [ ] |
| ( )|           |           |           | ( ) |
+-----+-----+-----+-----+-----+
| L  |     K |     J |     I |     H |
| [ ]| [UI] | [UP] | [UH] | [ ] |
| ( )| ( ) | ( ) | ( ) | ( ) |
+-----+-----+-----+-----+-----+
(15 ms) yes
```

Ketika perintah dijalankan :

```
| ?- restart_reversal.
Restart Reversal Technique berhasil!
Unta Biru (paling atas) dari petak C telah dikembalikan ke Start/Finish
(S/F).
```

Map sesudah

```
| ?- displayMap.
+-----+-----+-----+-----+-----+
| S/F |     A |     B |     C |     D |
| [UB]|     [ ]|     [ ]| [UM, UK] |     [ ] |
| ( )|     ( )|     ( )| ( )           |     ( ) |
+-----+-----+-----+-----+-----+
| O  |           |           |           |     E |
| [ ]|           |           |           | [ ] |
| ( )|           |           |           | ( ) |
+-----+-----+-----+-----+-----+
| N  |           |           |           |     F |
| [ ]|           |           |           | [ ] |
| ( )|           |           |           | ( ) |
+-----+-----+-----+-----+-----+
| M  |           |           |           |     G |
| [ ]|           |           |           | [ ] |
| ( )|           |           |           | ( ) |
+-----+-----+-----+-----+-----+
| L  |     K |     J |     I |     H |
| [ ]| [UI] | [UP] | [UH] | [ ] |
| ( )| ( ) | ( ) | ( ) | ( ) |
+-----+-----+-----+-----+-----+
yes .
```

KASUS GAGAL

Map sebelum dan sesudah perintah ini dijalankan tidak mengalami perubahan.

```
| ?- restart_reversal.
Restart Reversal Technique gagal!
Tidak ada perubahan posisi pada papan.

yes
```

4. Tukar Unta

Buatlah action `tukarUnta` yang menukar posisi 2 unta dengan warna acak (gunakan `kocokDadu()`). Unta-unta yang setumpuk dalam petak yang sama dapat ditukar posisi tumpukannya.

Contoh Program

```
?- tukarUnta().  
  
Kocok dadu...  
Warna unta 1: merah  
Warna unta 2: kuning  
  
Posisi sebelum unta merah: A  
Posisi sebelum unta kuning: L  
  
Unta merah dan kuning bertukar posisi...  
  
Posisi sesudah unta merah: L  
Posisi sesudah unta kuning: A
```

5. God's Hand

Tuhan berkuasa atas segala ciptaan-Nya. Tak terkecuali dengan unta di dalam papan permainan dan para pemain yang bertakwa. Seberapa jago dan beruntung seseorang, takdirnya tetap di tangan Tuhan.

Buatlah sebuah `action godsHand` yang mampu **memindahkan sebuah unta dari satu petak ke petak lain**. Jika ada tumpukan unta di petak tersebut, maka ambillah unta di tumpukan paling atas. `action` ini tidak dipanggil oleh pemain, melainkan terjadi tiap *turn* dengan probabilitas tertentu. Probabilitas tersebut dapat ditentukan secara bebas oleh masing-masing kelompok. `Action` ini mungkin saja menyentuh petak kosong. Sehingga, tidak ada unta yang dipindahkan.

Contoh Program

```
?-godsHand().
```

Tuhan telah berkehendak...
Dan dinamika petak berubah..

Unta Merah telah dipindahkan dari petak E ke petak A...

Peta:

S/F [UB UH] ()	A [UK, UM] ()	B ()	C ()	D ()
O ()				E ()
N [UP] ()		CAMEL UP!		F ()
M [UI] ()				G ()
L ()	K ()	J ()	I ()	H ()

yes

Penjelasan:

- Pada contoh ini, God's Hand pada turn ini berhasil memindahkan unta merah dari petak E ke petak C.
- Secara alami, petak yang dipindahkan bersifat random, jika tidak unta di petak tersebut maka God's Hand tidak mempengaruhi permainan di *turn* tersebut.
- God's Hand bisa menguntungkan ataupun merugikan pemain pada setiap *turn*-nya.

Kelompok

Pembagian kelompok ditentukan sendiri oleh mahasiswa dengan mengisi [sheet kelompok](#) berikut ini dengan 1 kelompok terdiri dari 4 hingga 5 mahasiswa yang berasal dari kelas yang sama. Batas waktu pengisian kelompok adalah **Rabu, 4 Desember 2024**. Silakan lihat instruksi lebih lanjut di sheet kelompok di atas.

FAQ

Pertanyaan dapat ditanyakan pada link [FAQ](#) berikut. Bacalah spesifikasi tugas secara lengkap terlebih dahulu sebelum bertanya. Pastikan pertanyaan yang ditanyakan tidak berulang. **Pertanyaan yang diajukan secara personal ke asisten tidak akan dijawab.**

Notula Asistensi

Notula untuk asistensi yang diadakan di kelas dapat diakses melalui link [berikut](#).

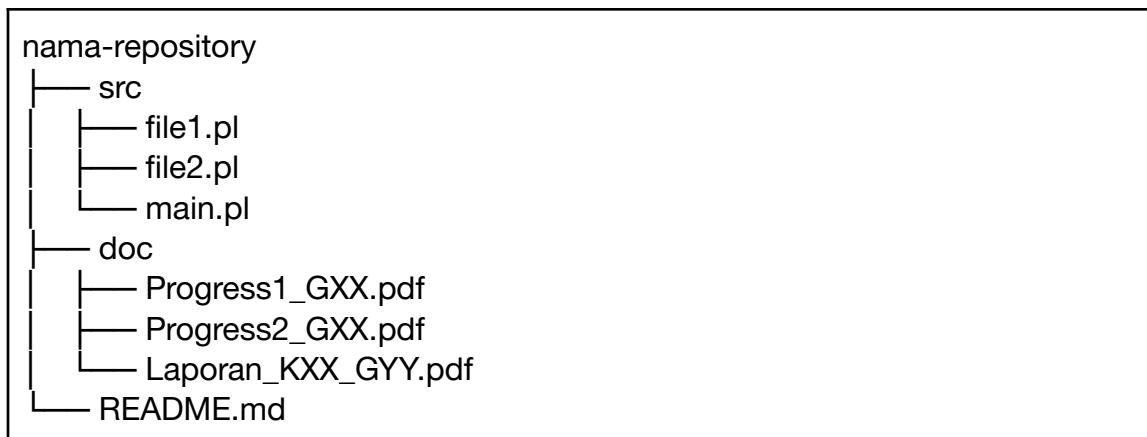
Aturan

Terdapat beberapa hal yang harus diperhatikan dalam penggerjaan tugas ini, yakni:

1. Query dan keluaran **tidak harus persis** dengan contoh program yang diberikan. Hal-hal yang tidak dispesifikkan dapat dianggap sebagai hal yang bebas diimplementasikan seperti apa, silakan berkreativitas sendiri.
2. Semua perubahan yang mungkin Anda lakukan diimplementasi Anda, ataupun fitur-fitur lain yang Anda ubah atau tambahkan, dapat dituliskan di laporan agar dapat dinilai oleh asisten dengan lengkap.
3. Apabila Anda mencari dan mencontoh kode dari Internet, harap cantumkan sumbernya dalam bentuk komentar dalam program dan referensi pada laporan.
4. Jika terdapat hal yang tidak dimengerti, silahkan ajukan pertanyaan kepada asisten melalui link FAQ yang telah diberikan di atas. Pertanyaan yang diajukan secara personal ke asisten tidak akan dijawab untuk menghindari perbedaan informasi yang didapatkan oleh peserta praktikum.
5. Dilarang melakukan plagiarisme, meminta/memberikan kode dan/atau laporan dari/ke kelompok lain, serta menggunakan generative AI dalam bentuk apapun. Pelanggaran pada poin ini akan menyebabkan pemberian nilai E pada setiap anggota kelompok yang melakukan maupun memberi.

Deliverables

1. Untuk tugas ini Anda diwajibkan menggunakan *version control system* git dengan menggunakan sebuah *repository private* di Github Classroom “IF2121 Logika Komputasional 2023” (gunakan surel *student* agar gratis).
2. Anda diwajibkan untuk membuat *repository* di dalam Github Classroom menggunakan link invitation berikut [ini](#). Berikut adalah alur untuk membuat *repository*:
 - a. Ketua kelompok mengisi kolom ‘NIM 1’ dan ‘Username GitHub 1’ pada [sheet kelompok](#) dan anggota lainnya mengisi kolom ‘NIM 2’ sampai ‘NIM 5’ beserta ‘Username GitHub 2’ sampai ‘Username GitHub 5’.
 - b. Semua anggota kelompok memilih *identifier* yang sesuai dengan kelas, NIM, dan nama mereka.
 - c. Ketua kelompok membuat *team* baru dengan nama kelompok yang sama dengan yang dicantumkan di [sheet kelompok](#).
 - d. Anggota kelompok melakukan *join* ke *team* yang sudah dibuat ketua kelompok.
3. Meskipun commit tidak dinilai, lakukanlah *commit* yang wajar dan sesuai *best practice* (tidak semua kode satu *commit*). Disarankan juga menggunakan [semantic commits](#) agar asisten lebih mudah menilai beban pekerjaan masing-masing anggota kelompok.
4. *Repository* memiliki struktur sebagai berikut.



5. Pengumpulan dilakukan dengan membuat *release* dengan tag vX.Y pada repository yang telah kelompok Anda buat sebelum deadline, dengan X adalah angka sesuai dengan nomor milestone dan Y adalah angka revisi dimulai dari 0 jika tidak ada revisi/versi awal. Pastikan tag sesuai format.
6. Khusus untuk milestone 3, pengumpulan dilakukan melalui edunex. Jika ada masalah pada edunex, pengumpulan dapat dilakukan melalui form [berikut](#) dengan **menyertakan bukti screenshot masalah pada edunex. Form akan ditutup tepat pada 20 Desember 2024 pukul 12.21.** Jika sudah mengumpulkan submisi di edunex, **tidak perlu mengisi form tersebut.**
7. Berikut adalah informasi tambahan mengenai Milestone dan *deliverables*-nya:
 - a. Pada Milestone 1, Anda diminta untuk paling tidak membuat detail lengkap **rencana fakta dan rule yang akan digunakan, baik yang menggunakan list, control loop.** Anda juga diminta untuk melampirkan **sebuah file berisi progress penggerjaan, rancangan rule, dan rancangan fakta** dengan format sebagai [berikut](#) dengan nama Progress1_GXX.pdf dengan XX merupakan nomor kelompok.
 - b. Pada Milestone 2, Anda diminta paling tidak sudah **mengimplementasikan fakta-fakta terkait permainan, implementasi rule-rule kendali dasar, dan start game** dalam bentuk **source code.** Anda juga diminta untuk melampirkan **sebuah file berisi progress penggerjaan keseluruhan** dengan format sebagai [berikut](#) dengan nama Progress2_GXX.pdf dengan XX merupakan nomor kelompok.
 - c. Pada Milestone 3, Anda diminta untuk **mengumpulkan berkas-berkas berikut** pada link [berikut](#).
 - i. Source code program keseluruhan
 - ii. Laporan hasil kerja dengan format penamaan Laporan_GXX.pdf dengan XX adalah nomor kelompok yang terdiri atas:
 1. Halaman cover yang memuat judul tugas, kode dan nama mata kuliah, dan identitas anggota kelompok

2. Penjelasan setiap *command* yang kelompok Anda telah buat, termasuk command bonus (jika ada), penjelasan meliputi:
 - a. Kegunaan command tersebut
 - b. Skenario-skenario penggunaannya (beserta contoh)
 - c. Tidak perlu menjelaskan cara kerja *command*
 3. Hasil eksekusi program berupa jalannya alur permainan (dalam bentuk *screenshot*)
 4. Pembagian dan persentase kerja masing-masing anggota kelompok
- Gunakan template [ini](#).
- iii. README file. Gunakan [*template*](#) ini untuk membantu penggerjaan kalian.
8. Penggerjaan tiap milestone akan **dinilai**.
 9. Tugas Besar IF1221 - Logika Komputasional adalah kegiatan yang bersifat mandiri berkelompok. Semua bentuk kecurangan akan ditindaklanjuti sesuai dengan sanksi akademik yang ada.

Referensi

https://www.tutorialspoint.com/prolog/prolog_tutorial.pdf

<https://docs.google.com/document/d/1t1URqqu2cMrDLkYdedUL6jRXinKrHyML98Yo8rBCnac/edit?tab=t.0#heading=h.gjdqxs>

<http://www.gprolog.org/>

Catatan Asisten



Ketika tidak ada bahu untuk bersandar, masih ada ~~tubes~~ batagor untuk didahar. -Qika

When you have lots of tubes but you are just a chill IF'23 student:





Kalo cape healing dulu ajah, chill bruhh😎 -eve

