

Spesifikasi Tugas Besar 1: Kompetisi Kartu ala Kerajaan Permen

IF2210 Pemrograman Berorientasi Objek

Revisi Terakhir: 13 Maret 2023 21:24 WIB

Deskripsi Persoalan



Sumber Gambar: https://adventuretime.fandom.com/wiki/Candy_Kingdom

"Pesta kerajaan permen."

Dalam rangka merayakan pesta panen gula di kerajaan permen, akan diadakan kompetisi *massal* permainan kartu ala Kerajaan Permen. Ratu kerajaan permen ingin mencari pemain kartu terhebat di kerajaannya. Namun karena Ratu tidak bisa memantau permainan satu-persatu, kalian diminta untuk membuat sistem agar permainan dapat berjalan dengan adil. Bantulah ratu kerajaan permen untuk membuatkan permainan ala kerajaan permen.

Alur Program

1. Program menginisiasi **7 pemain**. Dan secara bergiliran, player melakukan *input* nama/*nickname*. Setiap player akan memiliki poin 0 pada mula-mula.
2. Poin hadiah awalnya akan dibuat sejumlah 64. Poin ini akan dinamis bertambah/berkurang tergantung perintah dari setiap pemain. Poin akan diberikan diujung ronde (ronde 6).
3. Program akan mengeluarkan sebuah menu dan memberikan 2 opsi untuk melakukan generasi urutan kartu pada *deck* untuk permainan ini.
 - a. Opsi random
 - b. Opsi baca dari file
Untuk keperluan demo, sediakan opsi untuk memanipulasi urutan kartu pada *deck* dengan cara membaca teks dari sebuah file.
4. Program akan memulai permainan ini dengan putaran permainan yang dijelaskan pada bagian Mekanisme di poin Putaran Permainan.
5. Dalam permainan, pemain dapat memilih sebuah perintah, daftar perintah sebagai berikut:
 - a. Double
 - b. Next
 - c. Half
 - d. Ability (kemampuan)
6. Pada akhir permainan, poin hadiah akan diberikan ke pemenang. Dan pemain lain tidak mendapatkan poin. Permainan berakhir jika seorang pemain memiliki poin lebih dari sama dengan 2^{32} poin.
7. Jika belum ada yang menyentuh 2^{32} poin, maka permainan akan dilanjutkan dengan memulai dari ronde awal lagi. Kartu dikembalikan, pemain akan mendapat kartu kemampuan acak di ronde kedua dan seterusnya. Program juga akan mengeluarkan menu untuk memberikan opsi random/baca dari file untuk permainan selanjutnya.

Mekanisme

Program yang Anda buat harus memenuhi beberapa mekanisme. Berikut merupakan mekanisme yang harus Anda implementasikan di program Anda:

- Deck Kartu Kerajaan Permen

Main deck terdiri dari angka 1 sampai 13 untuk setiap warna Hijau, Biru, Kuning dan Merah. Sehingga terdapat 52 kartu secara total. Jika terdapat kesamaan angka, maka akan dilihat warnanya untuk melihat kartu mana yang lebih berharga. Urutan warna dari terendah ke tertinggi: Hijau, Biru, Kuning, dan Merah.

Selain itu, terdapat juga *deck ability card* yang berisi 7 kartu diantaranya. Total kartu 7, akan selalu habis karena akan dibagi rata ke 7 pemain yang ada, kartu akan dibagi pada ronde kedua.

1. Re-Roll
2. Quadruple
3. Quarter
4. Reverse Direction
5. Swap Card
6. Switch
7. Abilityless

- Player Card & Table Card

Pada permainan kartu, tiap pemain akan diberikan **dua buah Player Card**. Seorang pemain memiliki objektif untuk mendapatkan nilai kartu terbesar berdasarkan kartu yang dimiliki olehnya (Player Card) dan kartu yang ada di meja (Table Card). Nilai kartu disebut juga sebagai *hand* dan urutan nilai dapat dilihat lebih lanjut pada bagian "Combination" di bab "Mekanisme".

Table Card berjumlah hanya 1 objek untuk semua pemain, dan bisa dilihat oleh semua pemain. Table Card berisi 0 kartu dan akan ditambah 1 kartu setiap akhir ronde sampai berjumlah maksimal 5.

- Putaran Permainan

Terdapat 6 ronde dalam permainan. Pada setiap ronde, giliran pemain menggunakan sistem *round robin*. P1 ke P2 ke P3 dan seterusnya. Jika permainan belum selesai, sistem *round robin* dilanjutkan.

Permainan ke-	Ronde	Urutan
1	1	P1, P2, P3, P4, P5, P6, P7
	2	P2, P3, P4, P5, P6, P7, P1
	3	P3, P4, P5, P6, P7, P1, P2
 dan seterusnya
	5	P5, P6, P7, P1, P2, P3, P4
	6	P6, P7, P1, P2, P3, P4, P5
2	1	P7, P1, P2, P3, P4, P5, P6
	2	P1, P2, P3, P4, P5, P6, P7
 dan seterusnya

Catatan: Arah dapat berputar jika ada yang memakai kemampuan *reverse direction*.

1. Ronde pertama

Setiap pemain mendapat 2 kartu dari *deck utama* (angka dan warna). Pemain hanya bisa melihat kartu tersebut dan menentukan aksi: Double, Next, dan Half. Setelah ronde ini berakhir, setiap pemain akan diberikan sebuah kartu ability dan akan diletakkan sebuah kartu pada *table card* yang bisa dilihat oleh semua pemain.

2. Ronde kedua-kelima

Tidak berbeda dengan ronde sebelumnya, namun pemain dapat memakai opsi aksi Ability sesuai kartu yang didapat, kartu ini hanya bisa dipakai sekali dan **tidak wajib dipakai** sampai akhir permainan. Setelah ronde ini selesai, akan dibuka kartu kedua pada ronde kedua, kartu ketiga pada ronde ketiga, dan seterusnya.

Khusus ronde kedua, ada pembagian ada kartu *ability*.

3. Ronde keenam/terakhir

Sama seperti ronde sebelumnya, hanya saja tidak menambah kartu keenam pada *table card*.

4. Perhitungan kartu

Pemain akan membuka semua kartu miliknya, lalu akan dicari pemenang dengan melihat nilai tertinggi. Poin hadiah diberikan ke pemenang. Semua kartu *reset*, permainan diulang dari awal sampai ada pemain yang melewati skor 2^{32} .

Dalam satu kali ronde, setiap player hanya dapat melakukan satu kali giliran.

- Poin Hadiah

Setiap ronde 1 permainan, poin hadiah akan dimulai dengan angka nilai 64. Poin hadiah sifatnya sama untuk semua pemain. Melalui aksi pemain, nilai ini akan berubah dan akan diberikan kepada pemenang. Nilai poin paling kecil adalah 1. Sehingga, aksi apapun yang mengurangi poin lebih dari 1 tidak ada efeknya. Poin hadiah akan diberikan kepada pemain dengan nilai kartu tertinggi di akhir ronde ke 6.

- Perintah Double dan Half

Seorang pemain dapat melakukan perintah *double* untuk membuat poin hadiah berjumlah 2x atau 0.5x dari sebelumnya. Perlu diingat, apabila poin hadiah bernilai 1, maka perintah half tidak ada efeknya atau sama saja dengan perintah Next.

- Perintah Next

Tidak melakukan apa-apa dan giliran diteruskan ke pemain selanjutnya.

- Ability

Kartu Ability akan diberikan ke pemain pada ronde kedua. Pemain dapat menggunakan *ability* dan melihat kartu *ability* pada giliran kedua. Kartu *ability* hanya ada tujuh sehingga setiap permainan pasti akan habis dibagi ke semua pemain. Kartu *ability* hanya bisa dipakai sekali per permainan dan bisa juga tidak digunakan. Terdapat kekuatan diantaranya.

1. Re-Roll, membuang 2 kartu dari *main deck* yang dimiliki oleh diri sendiri dan mengambil ulang 2 kartu.
2. Quadruple, sama seperti aksi *double* hanya saja multipliernya menjadi 4x.
3. Quarter, sama seperti aksi *half* hanya saja multipliernya menjadi 0.25x.
4. Reverse Direction, memutar arah giliran, pemain yang sudah melakukan aksi pada giliran tersebut akan dilewati.

Contoh:

Urutan normal: P2 P3 P4 P5 P6 P7 P1

P2 melakukan aksi.

P3 melakukan aksi.

P4 menggunakan *ability reverse*.

Giliran diberikan ke P1, P7, P6 dan P5 pada ronde tersebut. (P2 dan P3 dilewati karena tadi sudah melakukan aksi)

Ronde berikutnya urutan menjadi P3 P2 P1 P7 P6 P5 P4, seharusnya P3 P4 P5 P6 P7 P1 P2.

5. Swap Card, menukar 1 kartu *main deck* milik pemain lain dengan 1 kartu *main deck* milik pemain lain. Tidak boleh ditukar dengan kartu *main deck* diri sendiri.
6. Switch, menukar 2 kartu *main deck* milik diri sendiri dengan 2 kartu *main deck* milik pemain lain. Harus bertukar milik sendiri dengan pemain lain. Tidak boleh ke 2 pemain lain.
7. Abilityless, mematikan kartu *ability* milik pemain lain. Harus digunakan saat giliran (**preventif**), tidak bisa mencegah seperti mematikan kartu *ability* ketika ada yang memakai. Sehingga, apabila semua pemain selain pemegang kemampuan ini sudah memakai kemampuan, maka secara tidak langsung pemegang kemampuan ini terkena *abilityless* (karena semua kartu kemampuan sudah dipakai) dan harus memilih aksi selain power.

● Combination

Urutan nilai kartu secara umum dari tertinggi ke terendah dilihat melalui angka. Apabila angka sama, maka akan dilihat melalui urutan warna, Merah > Kuning > Biru > Hijau.

Pada akhir permainan, akan dicari nilai tertinggi kombinasi kartu yang seorang pemain dapat buat. Terdapat berbagai kombinasi yang dapat muncul dengan urutan kekuatan yang berbeda. Sebuah kombinasi pasti kalah dengan kombinasi yang levelnya berada di atasnya. Namun, jika levelnya sama, akan diambil kartu dengan nilai tertinggi pada kombinasi tersebut. Untuk penjelasan kombinasi dari yang terlemah ke terkuat akan dijelaskan sebagai berikut.

1. High Card

High card merupakan kartu terbesar di *hand* pemain apabila tidak terdapat kombinasi lain. Apabila *high card* bernilai sama, bandingkan warna.

2. Pair

Pair terbentuk apabila terdapat 2 kartu dengan nilai yang sama. Jika terdapat >1 pemain yang memiliki *pair*, maka urutan *pair* akan dilihat terlebih dahulu dari angka. Jika terdapat kasus dua pemain memiliki angka *pair* yang sama, maka urutan *pair* akan dilihat melalui warnanya. Contoh:

Kasus 1

Pemain A = Pair 7 (Merah dan Hijau)

Pemain B = Pair 9 (Merah dan Biru)

Pemain B > Pemain A. Alasannya karena *pair* 9 lebih kuat dibanding *pair* 7.

Kasus 2

Pemain A = Pair 7 (Biru dan Merah)

Pemain B = Pair 7 (Hijau dan Kuning)

Pemain A > Pemain B. Alasannya walaupun mereka memiliki *pair* yang sama, tetapi pemain A memiliki warna terkuat (Merah).

3. Two Pair

Two pair terbentuk apabila terdapat 2 *pair* kartu pada *hand* milik pemain. Jika terdapat >1 pemain yang memiliki kombinasi 2 *pair*, maka urutan akan dilihat melalui *pair* besar. Contoh:

Kasus 1

Pemain A = Two Pair 10 (2, Hijau dan Kuning) 2 (2, Merah dan Biru)

Pemain B = Two Pair 5 (2, Merah dan Kuning) 4 (2, Hijau dan Kuning)

Pemain A > Pemain B. Karena $10 > 5$.

Kasus 2

Pemain A = Two Pair 10 (2, Hijau dan Merah) 2 (2, Merah dan Biru)

Pemain B = Two Pair 10 (2, Biru dan Kuning) 7 (2, Merah dan Biru)

Pemain A > Pemain B. Alasannya karena warna *pair* pada pemain A (Merah) lebih besar dibanding warna *pair* pada pemain B (Kuning).

4. Three of a Kind

Three of a kind terbentuk apabila terdapat 3 kartu dengan nilai yang sama. Jika terdapat >1 pemain yang memiliki kombinasi *Three of a kind*, maka urutan akan dilihat dari angka yang membentuk *Three of a kind* tersebut. Contoh:

Pemain A = Three of a Kind 8

Pemain B = Three of a Kind 11

Pemain B > Pemain A.

5. Straight

Straight terbentuk apabila terdapat 5 kartu yang berurutan dalam *hand* milik pemain. Jika terdapat >1 pemain yang memiliki kombinasi *straight*, maka urutan akan dilihat dari **angka terbesar** yang membentuk *straight* tersebut. Contoh:

Pemain A = Straight 2 3 4 5 **6**

Pemain B = Straight 4 5 6 7 **8**

Pemain B > Pemain A.

6. Flush

Flush terbentuk apabila terdapat 5 kartu dengan warna yang sama dalam *hand* milik pemain. Jika terdapat >1 pemain yang memiliki kombinasi *flush*, maka urutan akan dilihat dari **angka terbesar** yang membentuk *flush* tersebut. Contoh:

Kasus 1

Pemain A = Flush 13 11 9 6 4 (Hijau)

Pemain B = Flush 12 11 7 6 5 (Biru)

Pemain A > Pemain B. Alasannya karena pemain A memiliki angka 13 di dalam kombinasi flushnya. 13 > 12.

Kasus 2

Pemain A = Flush 13 11 9 6 4 (Hijau)

Pemain B = Flush 13 12 7 6 5 (Biru)

Pemain B > Pemain A. Alasannya karena pemain B memiliki angka 12 sebagai terbesar kedua setelah King dalam kombinasi flushnya.

Kasus 3

Pemain A = Flush 13 11 10 3 1 (Hijau)

Pemain B = Flush 13 11 10 3 1 (Biru)

Pemain B > Pemain A. Alasannya karena pemain B memiliki kartu berwarna biru. Biru > Hijau.

7. Full House

Full house terbentuk apabila terdapat 3 kartu dengan nilai yang sama dan 2 kartu lain dengan nilai yang sama. Apabila terdapat >1 pemain yang memiliki kombinasi *full house*, *hand* yang lebih besar diukur dari nilai 3 kartu yang memiliki nilai sama. Contoh:

Pemain A = Full House 12 (3) 3 (2)

Pemain B = Full House 7 (3) 13 (2)

Pemain A > B. Alasannya karena pemain A memiliki 3 buah angka 12, walaupun pemain B memiliki 2 buah 13 hal tersebut masih kalah karena yang dilihat terlebih dahulu adalah kartu yang berjumlah 3.

Catatan: tidak mungkin ada kasus memiliki full house dengan angka yang sama, karena sebuah *deck* hanya memiliki 4 kartu untuk angka yang sama.

8. Four of a Kind

Four of a kind terbentuk apabila terdapat 4 kartu dengan nilai yang pada *hand* milik pemain. Jika terdapat >1 pemain yang memiliki kombinasi *four of a kind*, maka urutan akan dilihat dari angka yang membentuk kombinasi *four of a kind*. Contoh:

Pemain A = Four of a Kind 7

Pemain B = Four of a Kind 11

Pemain B > Pemain A.

Catatan: tidak mungkin ada kasus memiliki four of a kind dengan angka yang sama, karena sebuah *deck* hanya memiliki 4 kartu untuk angka yang sama.

9. Straight Flush

Straight flush terbentuk apabila terdapat 5 kartu berurutan dengan warna yang sama pada *hand* milik pemain. Jika terdapat >1 pemain yang memiliki kombinasi *straight flush*, maka urutan akan dilihat dari angka lalu warna yang membentuk kombinasi *straight flush*. Contoh:

Kasus 1

Pemain A = Straight Flush 11 10 9 8 7 (Biru)

Pemain B = Straight Flush 12 11 10 9 8 (Hijau)

Pemain B > Pemain A. 12 > 11.

Kasus 2

Pemain A = Straight Flush 11 10 9 8 7 (Biru)

Pemain B = Straight Flush 11 10 9 8 7 (Hijau)

Pemain A > Pemain B. Alasannya karena warna straight flush pemain A adalah biru. Biru > Hijau.

Untuk memudahkan dalam implementasi program, kalian dapat mengikuti perhitungan bobot di bawah ini untuk menentukan urutan kombinasi:

	Konstan	Hijau	Biru	Kuning	Merah
1	0.1	0.1	0.13	0.16	0.19
2	0.2	0.2	0.23	0.26	0.29
3	0.3
4	0.4
5	0.5
6	0.6
7	0.7
8	0.8
9	0.9
10	1.0
11	1.1
12	1.2
13	1.3	1.3	1.33	1.36	1.39

Rumus Perhitungan *High Card* : $\text{Konstanta} + \text{Warna} * 0.03$

*dengan masing-masing warna memiliki nilai, hijau = 0, biru = 1, kuning = 2, merah = 3.

Rumus Perhitungan Combination :

- Pair = <rumus_menghitung_pair> + 1.39 (1.39 adalah nilai maksimum dari single card/high card)
- Two Pair = <rumus_menghitung_two_pair> + <nilai_maksimum_pair>

- Three of a Kind = $\langle \text{rumus_menghitung_three_of_a_kind} \rangle + \langle \text{nilai_maksimum_two_pair} \rangle$
- Straight = $\langle \text{rumus_menghitung_straight} \rangle + \langle \text{nilai_maksimum_three_of_a_kind} \rangle$
- Flush = $\langle \text{rumus_menghitung_flush} \rangle + \langle \text{nilai_maksimum_straight} \rangle$
- Full House = $\langle \text{rumus_menghitung_full_house} \rangle + \langle \text{nilai_maksimum_flush} \rangle$
- Four of a Kind = $\langle \text{rumus_menghitung_four_of_a_kind} \rangle + \langle \text{nilai_maksimum_full_house} \rangle$
- Straight Flush = $\langle \text{rumus_menghitung_straight_flush} \rangle + \langle \text{nilai_maksimum_four_of_a_kind} \rangle$

Silahkan masing-masing rumus kombinasi kalian cari sendiri berdasarkan urutan yang telah dijelaskan di atas. Carilah perhitungan yang menghilangkan kemungkinan *overlap* antar kombinasi. Letakkan rumus ini untuk abstraksi kalian menghitung nilai sebuah kartu atau kombinasi.

Kasus Unik Yang Perlu Diperhatikan

- Ketika kartu di meja sudah menunjukkan kombinasi tertinggi (*straight flush* tertinggi) dan tidak ada lagi pemain yang membuat *straight flush*, maka perhitungan siapa *player* yang menang masih ditentukan dengan kombinasi kartu di tangan *player*. Ingat, kombinasi yang mungkin diperhitungkan dalam kasus ini adalah *high card*, *pair*, *two pair*, atau *three of kind* (~~hayo tebak kenapa kombinasi lain tidak bisa diperhitungkan~~).
- Kasus lain: semisal di *tablecard* ada flush kuning, dan setiap player tidak memiliki kartu kuning, maka lihat kartu tertinggi dari kombinasi lain.
- Kasus lain 2: semisal di *tablecard* ada flush kuning, dan ada player yang memiliki kartu kuning di tangan, maka pemenangnya adalah flush kuning dengan kartu kuning tertinggi ditangan.
 Table Card: Kuning 3 5 7 9 11 13
 Player 1: Kuning 6, Merah 13
 Player 2: Kuning 8, Hijau 3
 Player 3: Pair 5 (Merah & Kuning)
 Pemenang: Player 2, karena mampu membuat Flush 8, (11 dan 13 dianggap sama karena setiap pemain punya kartu flush tersebut).
- Intinya, tidak ada kasus seri, selalu ambil kartu paling tertinggi, jika ada yang sama, bisa diignore atau coba dicari tertinggi dalam kesamaan tersebut. Karena pasti bisa ditemukan kartu paling tertinggi.

Operasi dan Command

Command harus diikuti sesuai format syntax yang ada, kecuali jika mengerjakan bonus.

1. Next

Perintah yang memiliki arti pemain tidak melakukan apa-apa (tidak Half, tidak Double, dan tidak mengeluarkan kartu *ability*). Giliran akan diserahkan kepada pemain berikutnya.

```
NEXT
Giliran dilanjut ke pemain selanjutnya.
```

Tabel 1 Melakukan perintah Next.

2. Re-Roll

Perintah yang memiliki arti pemain membuang kedua kartu yang ada di tangannya dan mengambil ulang dua kartu baru. Perintah ini hanya bisa dilakukan ketika pemain memiliki kartu kemampuan Re-Roll dan belum pernah digunakan. Setelah perintah dilakukan, giliran diberikan ke pemain selanjutnya.

```
RE-ROLL
Melakukan pembuangan kartu yang sedang dimiliki
Kamu mendapatkan 2 kartu baru yaitu:
  1. 2 Merah
  2. 4 Biru
// F.S: Giliran dilanjut.
```

Tabel 2 Melakukan perintah Re-Roll.

3. Double/Quadruple

Perintah yang memiliki arti pemain akan menaikkan total poin hadiah pada permainan menjadi dua kali lipat atau empat kali lipat. Perintah **Quadruple** hanya bisa digunakan apabila pemain memiliki kartu Ability yang sesuai.

```
> DOUBLE
<nama_pemain> melakukan DOUBLE! Poin hadiah naik dari
<poin_sebelumnya> menjadi <poin_sekarang>!
// F.S: Giliran dilanjut.

> QUADRUPLE
<nama_pemain> melakukan QUADRUPLE! Poin hadiah naik dari
<poin_sebelumnya> menjadi <poin_sekarang>!
// F.S: Giliran dilanjut.

> QUADRUPLE
Ets, tidak bisa. Kamu tidak punya kartu Ability QUADRUPLE.
// F.S: Giliran tetap pada pemain ini.
```

```
>
```

Tabel 3 Melakukan perintah Double/Quadruple.

4. Half/Quarter

Perintah yang memiliki arti pemain akan menurunkan total poin hadiah pada permainan menjadi setengah atau seperempat nilai awal. Perintah **Quarter** hanya bisa digunakan apabila pemain memiliki kartu Ability yang sesuai. Apabila poin hadiah bernilai 1, maka perintah half tidak ada efeknya.

```
> HALF
<nama_pemain> melakukan HALF! Poin hadiah turun dari
<poin_sebelumnya> menjadi <poin_sekarang>!

// F.S: Giliran dilanjutkan.

> QUARTER
<nama_pemain> melakukan QUARTER! Poin hadiah turun dari
<poin_sebelumnya> menjadi <poin_sekarang>!

// F.S: Giliran dilanjutkan.

> HALF
<nama_pemain> melakukan HALF! Sayangnya poin hadiah sudah bernilai
1. Poin hadiah tidak berubah.. Giliran dilanjutkan!

// F.S: Giliran dilanjutkan.

> QUARTER
Ets, tidak bisa. Kamu tidak punya kartu Ability QUARTER.
// F.S: Giliran tetap pada pemain ini.

>
```

Tabel 4 Melakukan perintah Half/Quarter.

5. Reverse

Perintah yang memiliki arti untuk memutar arah giliran eksekusi perintah oleh pemain. Pemain yang sudah melakukan aksi pada giliran tersebut akan dilewati. Perintah ini juga akan mempengaruhi urutan eksekusi perintah oleh pemain pada giliran selanjutnya. Apabila kemampuan berhasil dipakai, giliran tidak dilanjutkan ke pemain berikutnya, pemain yang mengeksekusi kemampuan ini masih dapat mengeksekusi satu perintah lain (DOUBLE/HALF/NEXT).

```
//CASE 1
> REVERSE
<nama_pemain> melakukan reverse!
(sisa) urutan eksekusi giliran ini : <p1> <p4> <p6> <p2>
urutan eksekusi giliran selanjutnya : <p3> <p7> <p2> <p6> <p4> <p1>
<p5>
```

<pre>//CASE 2 > REVERSE Oops, kartu ability reversemu telah dimatikan sebelumnya(. Silahkan lakukan perintah lain. //CASE 3 > REVERSE Ets, tidak bisa. Kamu tidak punya kartu Ability REVERSE. Silahkan lakukan perintah lain.</pre>
<pre>CASE 1 I.S. : Kartu ability reverse tidak dimatikan. Player yang menggunakan reverse adalah <p5>. Urutan eksekusi giliran ini : <p7> <p3> <p5> <p2> <p6> <p4> <p1> Urutan eksekusi giliran selanjutnya : <p3> <p5> <p2> <p6> <p4> <p1> <p7> F.S. : Kartu ability reverse dapat digunakan. Urutan eksekusi giliran ini : <p7> <p3> <p5> <p1> <p4> <p6> <p2> Urutan eksekusi giliran selanjutnya : <p3> <p7> <p2> <p6> <p4> <p1> <p5> Giliran tetap pada pemain sekarang. Dengan kata lain, ini hanya satu-satunya kemampuan yang dapat membuat pemainnya mengeksekusi dua buah aksi sekaligus.</pre>
<pre>CASE 2 I.S. : Kartu ability reverse dimatikan (ada pemain lain yang menggunakan abilityless). F.S. : Kartu ability reverse tidak dapat digunakan, dikeluarkan pesan kesalahan, tidak terjadi perubahan urutan eksekusi, dan giliran masih tetap pada pemain sekarang .</pre>
<pre>CASE 3 I.S. : Pemain tidak mempunyai kartu ability reverse. F.S. : Dikeluarkan pesan kesalahan dan giliran masih tetap pada pemain sekarang.</pre>

Tabel 5 Melakukan perintah Reverse.

6. Swap Card

Perintah yang memiliki arti menukar satu kartu pemain lain dengan satu kartu pemain yang lain. Pemain akan memilih dua pemain lain yang akan menukarkan masing-masing satu kartu secara acak (kartu yang ditukar tidak dapat dipilih secara manual).

<pre>> SWAPCARD //PREKONDISI //Kartu pemain_3: 3(Merah) && 8(Biru) //Kartu pemain_6: 4(Hijau) && 10(Biru) <nama_pemain> melakukan SWAPCARD. Silahkan pilih pemain yang kartunya ingin anda tukar: 1. <pemain_2></pre>
--

```

2. <pemain_3>
3. <pemain_4>
4. <pemain_5>
5. <pemain_6>
6. <pemain_7>
> 5
Silahkan pilih pemain lain yang kartunya ingin anda tukar:
1. <pemain_2>
2. <pemain_3>
3. <pemain_4>
4. <pemain_6>
5. <pemain_7>
> 2

Silakan pilih kartu kanan/kiri pemain_3:
1. Kanan
2. Kiri
> 2
Silakan pilih kartu kanan/kiri pemain_6:
1. Kanan
2. Kiri
> 1
//KONDISI FINAL
//Kartu pemain_3: 3(Merah) && 4(Hijau)
//Kartu pemain_6: 8(Biru) && 10(Biru)

```

CASE 2
I.S. : Kartu ability swap dimatikan (ada pemain lain yang menggunakan abilityless).
F.S. : Kartu ability swap tidak dapat digunakan. Giliran masih pada pemain ini.

Tabel 6 Melakukan perintah swap.

7. Switch

Perintah yang memiliki arti pemain akan menukar kartu *main deck* miliknya dengan kartu *main deck* milik pemain lain. Kartu yang ditukar tidak bisa dipisah-pisah alias harus 2 kartu sekaligus dan penukarannya juga hanya boleh melibatkan 1 pemain lain.

```

//CASE 1
> SWITCH
<nama_pemain> melakukan switch!
Kartumu sekarang adalah:
2 (Merah) && 4 (Biru)
Silahkan pilih pemain yang kartunya ingin anda tukar:
1. <pemain_2>
2. <pemain_3>
3. <pemain_4>
4. <pemain_5>
5. <pemain_6>
6. <pemain_7>

```



```

> 5
Kedua kartu <nama_pemain> telah ditukar dengan <pemain_5>!
Kartumu sekarang adalah:
13 (Hijau) && 13 (Merah)

//CASE 2
> SWITCH
Oops, kartu ability switchmu telah dimatikan sebelumnya :(.
Silahkan lakukan perintah lain.

```

```

CASE 1
I.S. : Kartu ability switch tidak dimatikan. Kartu main deck
nama_pemain adalah 2 merah dan 4 biru. Kartu main deck pemain_5
adalah 13 hijau dan 13 merah.
F.S. : Kartu ability switch dapat digunakan dan kartu main deck
nama_pemain menjadi 13 hijau dan 13 merah. Sedangkan kartu main
deck pemain_5 menjadi 2 merah dan 4 biru. Giliran dilanjut ke
pemain berikutnya.

```

```

CASE 2
I.S. : Kartu ability switch dimatikan (ada pemain lain yang
menggunakan abilityless).
F.S. : Kartu ability switch tidak dapat digunakan. Giliran masih
pada pemain ini.

```

Tabel 7 Melakukan perintah switch.

8. Abilityless

Perintah yang memiliki arti pemain akan mematikan kemampuan kartu lawan. Jika lawan yang dipilih sudah menggunakan kartu abilitynya, maka kartu ini tidak akan melakukan efek apapun, sehingga penggunaan kartu ini perlu diperhatikan.

```

//CASE 1
> ABILITYLESS
<nama_pemain> akan mematikan kartu ability lawan!
Silahkan pilih pemain yang kartu abilitynya ingin dimatikan:
  1. <pemain_2>
  2. <pemain_3>
  3. <pemain_4>
  4. <pemain_5>
  5. <pemain_6>
  6. <pemain_7>
> 5
Kartu ability <pemain_5> telah dimatikan.

//CASE 2
> ABILITYLESS
<nama_pemain> akan mematikan kartu ability lawan!
Silahkan pilih pemain yang kartu abilitynya ingin dimatikan:
  1. <pemain_2>
  2. <pemain_3>

```

<pre> 3. <pemain_4> 4. <pemain_5> 5. <pemain_6> 6. <pemain_7> > 5 Kartu ability <pemain_5> telah dipakai sebelumnya. Yah, sayang penggunaan kartu ini sia-sia 😞 //CASE 3 > ABILITYLESS Eits, kamu tidak punya kartunya 😞 //CASE 4 > ABILITYLESS Eits, ternyata semua pemain sudah memakai kartu kemampuan. Yah kamu kena sendiri deh, kemampuanmu menjadi <i>abilityless</i>. Yah, penggunaan kartu ini sia-sia </pre>
<pre> CASE 1 I.S. : Kartu ability pemain yang dipilih masih belum digunakan. F.S. : Kartu ability pemain yang dipilih tidak bisa digunakan. </pre>
<pre> CASE 2 I.S. : Kartu ability pemain yang dipilih sudah digunakan. F.S. : Muncul pesan error dan giliran dilanjutkan ke pemain selanjutnya. </pre>
<pre> CASE 3 I.S. : Pemain tidak mempunyai kartu ability <i>abilityless</i>. F.S. : Dikeluarkan pesan kesalahan dan giliran masih tetap pada pemain sekarang. </pre>
<pre> CASE 4 I.S. : Pemain lain sudah memakai kemampuan. F.S. : Muncul pesan error dan giliran dilanjutkan ke pemain selanjutnya. </pre>

Tabel 8 Melakukan perintah *abilityless*.

9. Akhir Permainan

Jika terdapat seorang pemain yang menyentuh 2^{32} poin (4294967296).

<pre> Permainan berakhir. Leaderboard: 1. Pemain_1: 4300000050 2. Pemain_2: 1000000 3. Pemain_3: 50 4. Pemain_4: 1 5. Pemain_5: 1 6. Pemain_6: 1 7. Pemain_7: 0 Permainan dimenangkan oleh Pemain_1. </pre>

Lanjut?

1. Main lagi
2. Exit

>

CASE 1

I.S: Permainan selesai dan diminta untuk bermain lagi

F.S: Minta ulang input nama pemain 1-7, setup, dst. Sama seperti menjalankan program ulang. Namun harus kontinu/tidak boleh pakai cara tutup program, ditunjukan untuk mengecek *destructor* yang kalian benar.

CASE 2

I.S: permainan selesai dan diminta untuk keluar

F.S: program berhenti.

Bonus

1. Spesifikasi wajib hanya meminta kalian untuk mengimplementasikan **satu** *generic* untuk kelas **atau** fungsi. Jika ingin mengerjakan bonus ini, kerjakan *generic* untuk dua-duanya: kelas **dan** fungsi. Untuk *use case* dibebaskan ke mahasiswa.
2. Dengan memanfaatkan flow dari program diatas, permainan bisa disubstitusi dengan permainan kartu lain seperti *capsa*, cangkul, dan *UNO*.

Jika ingin mengerjakan bonus ini, **pilih salah satu** tipe permainan kartu dan buatlah pada program yang sama dengan memanfaatkan kelas yang ada dan melakukan *extend* dari base kelas yang ada.

Program *object oriented* yang baik seharusnya tidak perlu mengubah banyak struktur program jika permasalahan serupa tapi tak sama. Jika kalian memanfaatkan OOP dengan baik, *effort* untuk mengerjakan bonus ini seharusnya tidak besar.

Selain dari poin-poin bonus yang ditetapkan di atas, Anda diperbolehkan untuk bebas berkreasi menambahkan fitur baru asal tidak mengubah fitur wajib.

Spesifikasi Sistem

Ketentuan Umum

Buatlah aplikasi berbasis **Command Line Interface (CLI)** untuk permainan ini dalam bahasa **C++**.

Ketentuan Teknis Minimal

Berikut adalah hal-hal yang **minimal** wajib diimplementasikan di aplikasi yang Anda buat. Diperbolehkan **menambah** dengan memperhatikan konsep-konsep OOP serta desain dari aplikasi kalian! Perlu dicatat, yang ditulis disini hanyalah minimal dan sangat besar kemungkinan kelompok kalian akan menambah abstraksi/konsep lainnya untuk membuat game ini.

1. Inheritance **dan** Polymorphism
Action, Ability atau *Command* untuk setiap perintah/kekuatan yang ada.
2. Exception **dan** Exception Handling
 - a. Validasi *input* user untuk angka
 - b. Validasi *input* user untuk *text file* yang berisi urutan kartu
 - c. Validasi lainnya jika diperlukan.
3. Function Overloading **dan** Operator Overloading
 - a. Penggunaan + dan - untuk menambah/menarik kartu player maupun *deck card*
 - b. Penggunaan >, <, dan == untuk membandingkan:
 - i. Kartu/kombo dengan kartu/kombo
 - ii. Player dengan player
4. Abstract Class **dan** Virtual Function
 - a. Sebuah kelas abstrak yang memiliki *virtual function* *value()* yang akan *di-inherit* oleh kelas Kombo dan Kartu.
 - b. Sebuah kelas abstrak berupa *InventoryHolder* yang akan diturunkan menjadi Player dan *Table Card*
 - c. Sebuah kelas abstrak *ability* yang dapat melakukan suatu efek.
5. Generic Function **atau** Generic Class
 - a. Generic function untuk T untuk yang menerima array T dan mengembalikan T dengan nilai terbesar. T minimal Player, Kombo, dan Kartu.
6. STL (*Standard Template Library*)

Bebas dipakai pada kelas apapun yang mendukung pembuatan *game*. Wajib bersifat krusial ke struktur data dan mekanisme permainan, bukan hanya untuk *print* atau diletakkan pada sebuah kelas namun tidak terpakai. **Yang harus dipakai: Vector, Pair, dan Map.** Boleh tambah pakai yang lain jika perlu.

Perlu diperhatikan kata **"dan"** artinya harus keduanya, sedangkan kata **"atau"** berarti minimal salah satu saja.

Panduan Pengerjaan

Tugas Besar ini dikerjakan secara sendiri/berkelompok (*se-nyaman kalian mengerjakan*) yang berisi **4 sampai 5 anggota (6 anggota diperbolehkan untuk kampus Jatinangor)**. Silakan mendaftarkan kelompok di **IF2210 OOP - Kelompok**, paling lambat Sabtu, 25 Februari 2023 23:59 WIB. Mohon diperhatikan poin berikut.

1. Kelompok yang terbentuk tetapi tidak terdata pada *sheet* tersebut tidak akan dianggap kelompok.
2. Diperbolehkan lintas kelas namun **tidak lintas kampus**.
 - a. Agar memudahkan jadwal demo.
3. Anggota yang tidak terdaftar pada *sheet* kelompok tersebut akan menandakan bahwa anggota tersebut tidak termasuk kelompok tersebut.
 - a. Tidak boleh berpindah kelompok setelah masa deadline pendataan kelompok.
 - b. Tidak boleh ada di dua/lebih kelompok.

Kami tulis 2 poin ini rasanya gak penting tapi kemarin pernah ada kejadian ini. Bukan angkatan kalian kok.
4. Perhatikan juga **README** pada *sheet* kelompok tersebut.

Adapun ketentuan lain seperti berikut.

1. Wajib bisa dijalankan di sistem operasi **LINUX** dengan G++ versi 11, 14 atau 17.
 - a. Akan ada **pengurangan nilai** jika terjadi anomali seperti **output** yang tidak muncul karena sejauh pengerjaan kalian dikerjakan di **Windows**.
 - b. Berhak tidak dinilai jika program **tidak jalan total** atau gagal mensimulasikan **mayoritas kondisi** untuk *test case* yang dibuat asisten.
 - c. Windows Subsystem Linux atau **WSL** diperbolehkan untuk substitusi sistem operasi **LINUX**.
 - d. Dihimbau untuk setiap kelompok mengerjakan di **WSL** atau **Linux** daripada diakhir saat demo tidak jalan sama sekali/ada kasus unik.

Kemarin ketika tugas besar Algoritma Struktur Data banyak sekali anomali yang menyusahkan asisten saat demo, jadi lebih baik semua anggota mengerjakan di WSL/Linux dari awal.
2. Hanya diperbolehkan menggunakan *library* umum dan STL. Tuliskan justifikasi *library* yang dipakai pada laporan anda.
3. Tidak boleh plagiat dari internet maupun kelompok lain.
4. Usahakan penggunaan static, reference passing, kelas dan method lebih banyak (~75%) **dibandingkan** dengan penggunaan *global variable*, *pointer passing*, *fungsi* dan *prosedur*.

Pengecualian **penggunaan struct**, tidak diperkenankan sama sekali, silakan gunakan kelas dan objek.

Aturan ini jelas karena kalian mengerjakan tugas besar mata kuliah pemrograman berorientasi objek.

5. Dekomposisi yang baik dan implementasi yang tidak terlalu kompleks (sebuah method tidak terlalu panjang). Pecah-pecah dan buat method baru agar tidak terlalu kompleks.
6. Pastikan *destructor* kalian berjalan dengan baik (dealokasi *array* dan sebagainya) ketika objek dihancurkan.

Selain penilaian dari tes fungsional, akan dilakukan penilaian dari segi desain program. Untuk mendapatkan nilai yang maksimal dari segi desain, terapkanlah prinsip:

- **Hindari menggunakan 6 konsep OOP diatas hanya karena diwajibkan saja! Gunakanlah konsep OOP yang sesuai dengan kasusnya!**
- DRY (Don't Repeat Yourself), tidak memiliki kode duplikat, pindahkan ke fungsi/*class*.
- Memiliki struktur kelas yang mudah dipahami
- Mengikuti prinsip **SL** (S dan L-nya saja) di [SOLID](#)
 - S: Single Responsibility, sebuah kelas hanya memiliki **satu** tanggung jawab.

Pembuatan kelas yang paradigmanya masih sama seperti prosedural atau fungsional akan terkena *penalti berat*.

Contoh **yang akan terkena *penalti berat***:

- Membuat tugas besar ini dengan **sebuah kelas** dan *main program*.
- Program hanya terdiri dari beberapa kelas yang **tidak jelas tanggung jawabnya apa** seperti membuat kelas dengan nama anggota kelompok dan kelas merepresentasikan pekerjaan anggota kelompok tersebut. **Tanggung jawab kelas harus berorientasi ke kelas tersebut, bukan siapa yang membuat kelas tersebut.**
- Melakukan *obfuscation* terhadap nama kelas dan *method* seperti membuat kelas dan method dengan nama yang tidak jelas. Contohnya: Kelas Foo dan Kelas Bar memiliki *method* a(), b(), c() dst.
- Tidak mengkomposisikan objek, method, dan kelas dengan baik. Hanya menggunakan konstruktor kelas untuk melakukan sesuatu. Atau mungkin semua kelas pada program hanya punya method **run**. Dan main program memanggil **run** tersebut 1 per 1.
- Membuat banyak kelas, namun kebanyakan kelas yang ada hanya sebagai tempat manipulasi dan simpan data. Semua logik permainan **tidak boleh** diletakkan di sebuah atau beberapa kelas (ini sama saja seperti membuat kelas dewa). Pastikan kelas dan objek yang kalian buat ikut berpartisipasi dalam mekanisme permainan. Tidak boleh **semua kelas** hanya berisi setter/getter saja.
- L: Liskov Substitution Principle, sebuah kelas turunan harus dapat merepresentasikan *parent class*-nya.


Panduan Laporan

Sebagai programmer yang baik, Anda dilatih tidak hanya untuk membuat kode, tetapi juga merancang sehingga program kalian *maintainable*.

Dokumentasikan desain program Anda melalui laporan singkat dengan format **pdf** yang **wajib** berisi:

1. Class diagram program anda (lengkap dengan header method dan atribut kelas secara singkat).
2. Penggunaan 6 konsep OOP, tuliskan semua.
3. Pada spek diatas, ada beberapa poin yang detail implementasinya dibebaskan ke anda. Tuliskanlah teknik implementasi yang ada pilih beserta alasannya!
4. Justifikasi *library* umum dan STL (selain wajib) yang dipakai
5. Bonus yang dikerjakan
6. Fitur/Kreasi tambahan jika ada
7. Pembagian Tugas & Foto Kelompok

Asistensi

1. Asistensi sinkron wajib dilakukan minimal **satu kali**. Metode asistensi sinkron dibebaskan kepada kesepakatan asisten dan kelompok.
2. Asistensi digunakan untuk bertanya mengenai spesifikasi maupun pengerjaan yang sekiranya belum terbayang dan belum jelas.
3. Silakan melakukan pengisian pemilihan asisten melalui sheet berikut  (Choose your own Asisten) Tugas Besar 1 IF2210 PBO maksimal 25 Februari 2023 23:59 WIB.
4. Disediakan asistensi secara asinkron (opsional) dengan bertanya kepada asisten secara langsung melalui *LINE*. Apabila pertanyaan terlalu sering ditanya, akan dimasukkan pada bagian di bawah ini.

Frequently Asked Question

1. Apakah diperbolehkan menggunakan `<cmath>` dan `<cstring>`?
Diperbolehkan.
- 2.

Akan diupdate secara berkala.

Pengumpulan

1. Softcopy laporan dan source code program dikumpulkan pada [LINK FORM](#). Submission **ditutup hari Kamis, 16 Maret 2023 22:10 WIB**. Di luar waktu pengumpulan tersebut, deliverables tidak akan diterima.
2. Untuk penyusunan Laporan Tugas Besar, akan disediakan template laporan. Laporan dikumpulkan dalam format pdf.
3. Spesifikasi pengumpulan yaitu:
 - a. **1 kelompok 1x pengumpulan saja!** Jika ada revisi, silakan submit ulang, akan dipakai submisi paling terakhir.
 - b. File source code (semua modul dan program utama) dikompres ke dalam zip.
 - c. Hapus semua file *executable* atau *binary* dari zip pengumpulan.
 - d. File diberi nama "IF2210_TB1_XXX.zip" dengan penjelasan:
 - i. XXX: Kode Unik Kelompok (ditulis dalam 3 huruf) yang ada pada Sheet Kelompok
 - ii. Format untuk nama PDF Laporan sama seperti nama zip, ekstensinya saja yang diubah menjadi .pdf
 - e. Contoh : "IF2210_TB1_XYZ.zip" adalah file untuk kelompok XYZ.

Penilaian

Penilaian Tugas Besar 1 akan dilakukan oleh asisten sesuai kriteria yang tertera pada dokumen Spesifikasi Tugas Besar 1.

Secara umum, penilaian akan sama seperti tugas besar sebelumnya (Algoritma Struktur Data) dengan mensimulasikan *test case* dari Asisten. Pastikan hal-hal krusial yang dilakukan untuk simulasi tidak rusak. Contoh: Membaca urutan kartu dari *deck*.

Pastikan sebelum pengumpulan, submisi yang kalian kumpulkan sudah benar. Asisten berhak untuk tidak menilai apabila program tidak jalan meskipun kesalahan pengumpulan sesuai prosedur. **Pastikan juga program sudah memenuhi ketentuan yang sudah dituliskan diatas.**

Asisten mengharapkan kejujuran dari setiap mahasiswa, segala bentuk kecurangan dalam pengerjaan tugas besar baik kode maupun dokumen akan mendapatkan nilai E pada mata kuliah IF2210 - Pemrograman Berorientasi Objek.