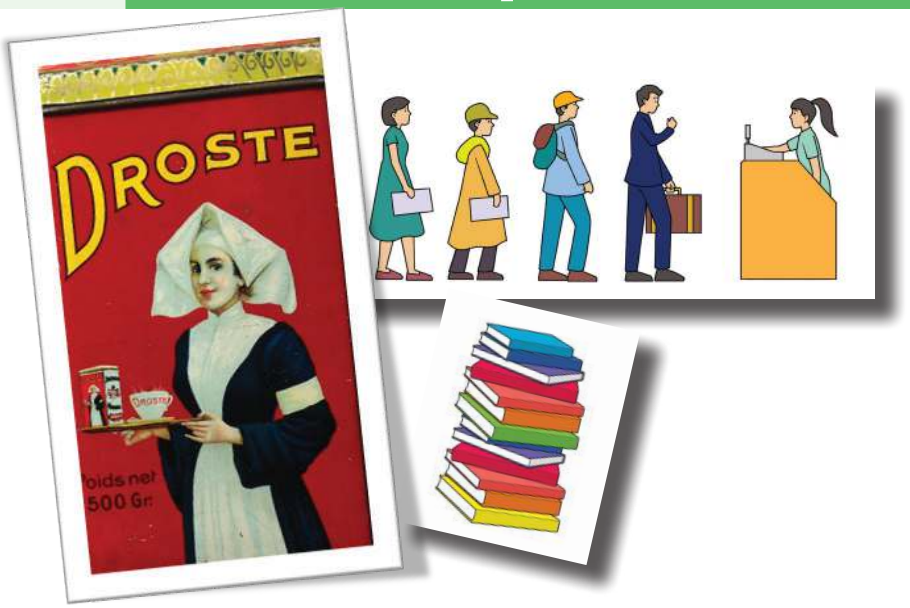
KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2021

Informatika untuk SMA Kelas X

Penulis : Mushthofa, Auzi Asfarian, Dean Apriana Ramadhan,
Irya Wisnubhadra
ISBN : 978-602-244-506-7

Bab 2

Berpikir Komputasional



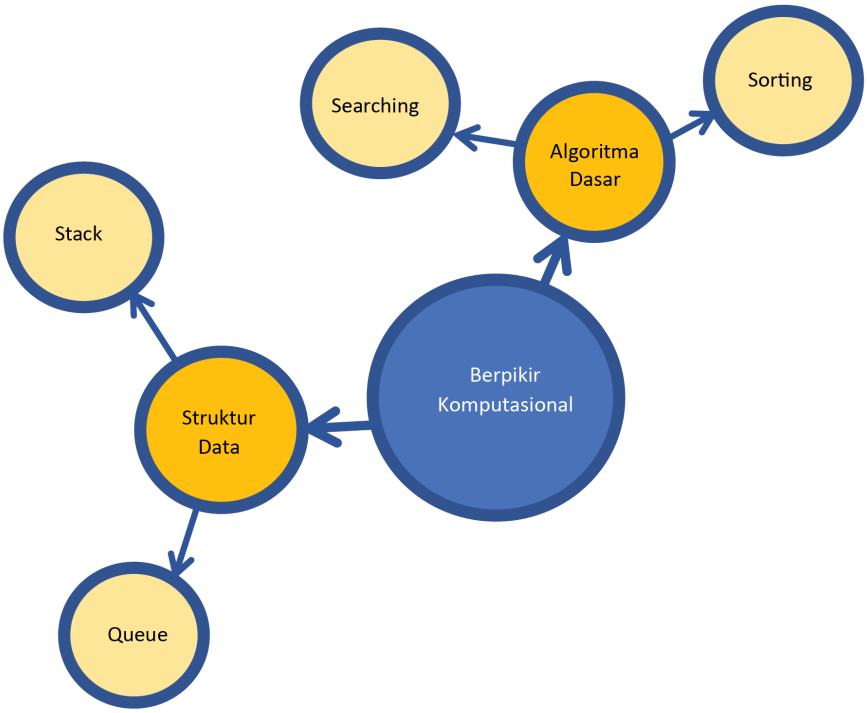
Tujuan Pembelajaran

Pada bab Berpikir Komputasional, kalian mampu menjelaskan dan menerapkan algoritma standar untuk beberapa persoalan yang disajikan, menjelaskan bagaimana data disimpan dalam struktur data tertentu, dan menentukan strategi yang efektif untuk menyelesaikan persoalan yang sajikan.

Pertanyaan Pemantik

Ketika kalian menghadapi suatu permasalahan yang harus kalian selesaikan, tentunya kalian menentukan strategi, bukan? Apa itu strategi? Mengapa dalam menemukan solusi, kalian perlu menentukan strategi?

Peta Konsep



Gambar 2.1 Peta Konsep Berpikir Komputasional

Apersepsi

Banyak peralatan dan aplikasi di dunia digital sekarang ini menggunakan komputer. Komputer dan aplikasinya dibuat oleh ahli Informatika sehingga membuat komputer tampak seperti mesin yang cerdas. Namun, benarkah komputer itu cerdas? Bagaimana komputer bisa melakukan tugas seperti manusia?

Kata Kunci

Algoritma, strategi algoritmik, *searching, sorting, stack, queue*.

Apa itu Berpikir Komputasional?

Melalui Berpikir komputasional (BK), kalian akan berlatih berpikir seperti seorang ilmuwan Informatika, bukan berpikir seperti komputer karena komputer adalah mesin.

Kegiatan utama dalam BK ialah penyelesaian masalah (*problem solving*), untuk menemukan solusi yang efisien, efektif, dan optimal sehingga solusinya bisa dijalankan oleh manusia maupun mesin. Dengan kata lain, kegiatan dalam BK ialah mencari strategi untuk mengatasi persoalan. Persoalan apa yang akan diselesaikan? Sebetulnya, hampir semua persoalan sehari-hari mengandung konsep komputasi sehingga bisa diselesaikan dengan bantuan mesin komputer. Sebagai contoh, robot yang bertugas melayani penjualan di restoran atau mengantar makanan dan obat untuk pasien di rumah sakit yang sudah dipakai di beberapa negara maju, sistem komputer untuk memantau perkebunan sawit yang siap panen dan sebagainya. Sistem komputer pada pada hakikatnya meniru dunia ini untuk dijadikan dunia digital sehingga bisa membantu atau menggantikan manusia dalam melakukan pekerjaan-pekerjaan yang sulit maupun membosankan.

Ada 4 fondasi berpikir komputasional yang dikenal dalam ilmu Informatika, yaitu Abstraksi, Algoritma, Dekomposisi, dan Pola, yang sangat mendasar dan secara garis besar dijelaskan sebagai berikut.

1. *Abstraksi*, yaitu menyarikan bagian penting dari suatu permasalahan dan mengabaikan yang tidak penting sehingga memudahkan fokus kepada solusi.
2. *Algoritma*, yaitu menuliskan otomasi solusi melalui berpikir algoritmik (langkah-langkah yang terurut) untuk mencapai suatu tujuan (solusi). Jika langkah yang runtut ini diberikan ke komputer dalam bahasa yang dipahami oleh komputer, kalian akan dapat "memerintah" komputer mengerjakan langkah tersebut.
3. *Dekomposisi* dan formulasi persoalan sedemikian rupa sehingga dapat diselesaikan dengan cepat dan efisien serta optimal dengan menggunakan komputer sebagai alat bantu. Persoalan yang sulit apalagi besar akan menjadi mudah jika diselesaikan sebagian-sebagian secara sistematis.
4. Pengenalan *pola* persoalan, generalisasi serta mentransfer proses penyelesaian persoalan ke persoalan lain yang sejenis.

BK perlu diasah dengan latihan rutin, mulai dari persoalan sederhana dan kecil. Kemudian, secara bertahap, persoalannya ditingkatkan menjadi makin besar, kompleks, dan rumit. Makin besar dan kompleks suatu persoalan, solusinya makin membutuhkan komputer agar dapat diselesaikan secara

efisien. Pada tingkat SD dan SMP, strategi penyelesaian persoalan belum secara khusus dirumuskan dalam bentuk algoritma. Pada tingkat SMA, kalian akan belajar bagaimana caranya agar solusi masalahnya bisa dituliskan dalam bentuk algoritma yang efisien dan siap dibuat menjadi program komputer.

Ruang permasalahan di dunia ini luas sekali, dan tentunya tak seorang pun ingin hidupnya menghadapi persoalan. Setiap bidang juga mempunyai persoalan dari sudut pandang bidang masing-masing, dan akan mengusulkan penyelesaian dengan menggunakan konsep dan prinsip keilmuan bidangnya. Kita belajar dari persoalan-persoalan yang ada dan pernah diusulkan solusinya. Oleh karena itu, belajar penyelesaian persoalan ialah belajar dari kasus-kasus dan solusinya. Namun, persoalan yang dibahas itu perlu di adaptasi dengan konteks kita. Kita perlu membentuk pola persoalan dan pola solusi dari latihan penyelesaiannya. Karena sangat banyak, latihan persoalan perlu dipilih. Topik yang dipilih dalam BK untuk SMA dalam mata pelajaran Informatika merupakan persoalan-persoalan mendasar terkait kehidupan sehari-hari yang perlu dikuasai dan mengandung konsep Informatika yang dominan. Bisa saja persoalan tersebut berkaitan dengan bidang lain, tetapi kita akan fokus ke aspek informatika. Memusatkan penyelesaian persoalan dari satu sudut pandang ini merupakan berpikir kritis! Dengan mempelajari dan membahas latihan-latihan pada unit pembelajaran ini, diharapkan kalian akan mendapatkan dasar pengetahuan yang diperlukan untuk menemukan solusi-solusi yang membutuhkan program komputer. Melalui kasus yang dibahas, kalian diharapkan dapat membentuk katalog solusi, yang saat dibutuhkan, akan tinggal dipakai. Melalui kegiatan BK ini, kalian menabung potongan solusi yang kelak dapat dirangkai menjadi pola solusi yang dibutuhkan untuk persoalan nyata yang dihadapi.

A. Pencarian (*Searching*)

Hidup adalah pencarian yang tiada henti. Mari, kita berpikir ke pengalaman "mencari" dalam kehidupan sehari-hari. Perhatikan contoh berikut.

1. Pernahkah kalian merasa kebingungan saat mencari sebuah buku di lemari buku kalian? Atau bahkan di perpustakaan? Saat kalian meminta bantuan kepada petugas perpustakaan, mengapa dia dapat menemukan buku yang kalian cari dengan waktu yang lebih singkat?

2. Suatu hari, kalian kehilangan baju seragam yang harus dipakai pada hari itu dan kalian mencarinya. Apa strategi kalian supaya baju tersebut cepat ditemukan?
3. Kalian mengingat sebuah potongan lirik lagu, tetapi tidak ingat judul lagu tersebut. Bagaimana kalian bisa menemukan lagu tersebut dengan cepat?

Apa itu *mencari*? Mencari adalah menemukan "sesuatu" yang bisa berupa benda, angka, konsep, informasi yang memenuhi kriteria tertentu dalam suatu ruang pencarian. Masalah pencarian sangat umum ditemukan di dalam kehidupan, termasuk dalam dunia komputasi. Ketika melakukan suatu pencarian, kalian harus menemukan suatu benda atau objek yang memenuhi kriteria tertentu dari sekumpulan benda atau objek lain. Beberapa contoh dari masalah pencarian yang sering kalian temui ialah sebagai berikut.

1. Mencari buku dengan judul tertentu di rak buku perpustakaan.
2. Mencari pakaian batik seragam kalian di lemari yang berisi semua pakaian yang kalian miliki.
3. Mencari dokumen atau web tertentu dengan mesin pencari seperti Google.

Mencari benda nyata gampang, tinggal kita lihat dan kita cocokkan dengan mata. Namun, mencari informasi atau konsep yang tidak kelihatan? Hmmmmm... Tidak mudah!

(a) (b)

Gambar 2.2. Pencarian (a) buku di perpustakaan, (b) informasi di internet

Masalah pencarian dapat dibuat dalam bentuk yang lebih formal agar dapat diterapkan pada banyak kasus. Elemen pada masalah pencarian meliputi hal-hal berikut.

1. Sekumpulan benda atau objek.
2. Kriteria dari benda atau objek yang dicari.
3. Pengecekan benda atau objek, untuk memeriksa apakah ia memenuhi kriteria pencarian.

Pertanyaan selanjutnya ialah bagaimana strategi untuk mencari. Banyak cara yang dapat kita lakukan, misalnya: kita dapat mengambil pakaian secara acak dan mengecek apakah pakaian tersebut ialah seragam batik. Cara lain, misalnya dengan memeriksa pakaian dari yang berada paling atas ke paling bawah. Tentunya, ada banyak strategi lain yang dapat kalian gunakan. Ada strategi yang lebih baik daripada strategi yang lain, bergantung pada keadaan benda atau objek tersebut saat pencarian dilakukan. Tentunya, kita akan lebih mudah mencari suatu buku dengan judul tertentu di lemari perpustakaan yang tersusun rapi dengan aturan tertentu dibandingkan dengan mencarinya di sebuah lemari yang berantakan.

Aktivitas Berpasangan

Aktivitas Bk-K10-01-U: Tebak Angka

Untuk memahami masalah pencarian, kalian akan bermain tebak angka. Pada saat bermain, cobalah untuk memahami permainan tersebut dan identifikasi aspek-aspek masalah pencarian pada permainan tersebut. Carilah strategi terbaik untuk menemukan angka yang dimiliki oleh teman kalian dengan jumlah pengecekan sesedikit mungkin.

Skenario Permainan

Pada permainan ini, kalian harus berpasangan dengan salah seorang teman. Teman kalian akan memilih sebuah angka bilangan bulat antara 1 – 100 (inklusif, angka 1 dan 100 juga boleh dipilih), dan angka tersebut akan ia rahasiakan. Tugas kalian ialah menemukan angka tersebut.

Untuk menemukan angka tersebut, kalian harus mengecek apakah angka tebakan kalian ialah angka yang dimiliki oleh teman kalian. Kalian hanya bisa mengecek angka satu per satu dengan menyebutkan angka tebakan kalian tersebut.

Setiap kali kalian menebak, teman kalian harus menjawab satu dari tiga kemungkinan berikut.

1. “Benar” apabila angka yang kalian tebak sama dengan angka yang dimiliki teman kalian.
2. “Angka milikku lebih kecil” apabila angka yang dimiliki teman kalian lebih kecil dari tebakan kalian.
3. “Angka milikku lebih besar” apabila angka yang dimiliki teman kalian lebih besar dari tebakan kalian.

Tentu saja, kalian dapat menebak angka apa pun, tetapi carilah strategi yang membuat kalian dapat dengan cepat (atau dengan kata lain jumlah tebakan sesedikit mungkin) menemukan angka yang dipilih oleh teman kalian.

Catatlah angka-angka yang kalian tebak dan jumlah tebakan yang kalian lakukan di lembar kerja yang disediakan. Lakukan permainan ini minimal sebanyak dua kali. Pada permainan berikutnya, kalian bisa bertukar peran.

Ilustrasi

Berikut ini ilustrasi dari permainan di atas. Pada permainan ini, Andi dan Binti bermain secara berpasangan. Andi memilih angka, sedangkan Binti harus menebak angka tersebut.

Mula-mula, Andi memilih sebuah angka antara 1 s.d. 100 (inklusif) dan Binti mencatat tebakannya di lembar kerjanya.

Binti: "50?"
Andi: "Angka milikku lebih kecil."
Binti: "30?"
Andi: "Angka milikku lebih besar."
Binti: "40?"
Andi: "Angka milikku lebih kecil."
Binti: "35?"
Andi: "Benar!"

Pada percakapan di atas, terlihat bahwa Binti dapat menebak angka yang dipilih Andi dalam empat kali penebakkan. Percakapan tersebut dicatat dalam lembar kerja, pada permainan ke-0.

Kalian dapat memanfaatkan tabel berikut yang digunakan untuk mencatat angka yang ditebak dan berapa kali penebakan dilakukan. Pada permainan ke-0, diberikan contoh isian dari ilustrasi yang diberikan di atas. Terlihat bahwa Andi memilih angka 49. Binti memerlukan sebanyak 4 kali penebakan untuk menebak dengan benar angka yang dipilih oleh Andi, yaitu menebak secara berurutan 50, 40, 35, dan 35.

Jawabannya tentu bergantung pada tebakan berikutnya.
Siapa Andi? .....
Siapa Binti? .....

Permainan ke-	Penebakan ke-																	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	50	40	30	35														
2																		
3																		

Apa yang kalian diskusikan?

Setelah bermain, saatnya kalian memikirkan makna permainan tersebut dan cara kalian bermain. Beberapa poin diskusi yang akan kalian lakukan seperti berikut.

1. Apakah permainan ini merupakan masalah pencarian?
2. Apabila Binti menjalankan strategi yang tepat, berapa kali jumlah maksimal tebakan yang benar-benar ia perlukan?
3. Strategi pencarian seperti apa yang kalian lakukan untuk menebak sesedikit mungkin?
4. Apakah strategi kalian berbeda dengan strategi yang dilakukan teman kalian? Jika berbeda, apa perbedaannya?
5. Strategi paling bagus apa yang dapat kalian temukan untuk menemukan angka dengan jumlah tebakan paling sedikit?
6. Adakah cara lain untuk "mencari" angka yang ditebak?

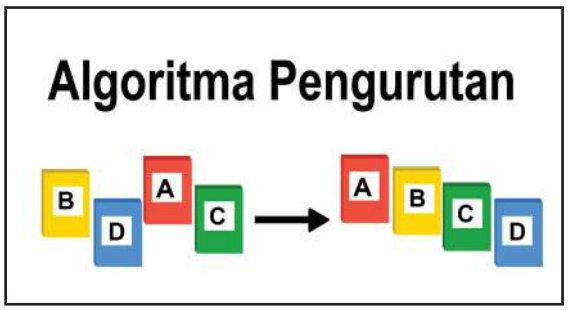
Apa yang kalian lakukan?

Tuliskan algoritma Tebak Angka dalam bahasa Indonesia. Masukkan dalam Buku Kerja Siswa.

Tantangan

Simpan tantangan ini sampai kalian sudah bisa memprogram dari modul Pemrograman. Jika kalian terpaksa melakukan permainan tebak angka sendiri karena belajar di rumah, ayo, lakukan hal ini bersama sebuah program komputer! Program komputer akan menjadi Andi, angka yang dipikirkan akan ditentukan secara acak oleh komputer. Kalian akan bisa membuat program komputer yang menjadi Andi jika kalian sudah belajar modul pemrograman. Catat hal ini sebagai salah satu hal yang akan kalian lakukan!

B. Pengurutan (*Sorting*)



Saat merapikan sesuatu, misalnya koleksi buku, kita menyusun buku tersebut dengan menggunakan suatu aturan. Misalnya, jika kita memiliki koleksi buku cerita berseri, kemungkinan besar kita akan menyusunnya secara berurut dari volume pertama hingga volume yang terbaru. Atau, ketika sedang berbaris, kita diminta untuk membentuk barisan berdasarkan tinggi badan. Hal-hal tersebut merupakan sebuah proses pengurutan atau sorting. Proses pengurutan akan

menjadi bagian yang tidak terpisahkan dari program komputer atau aplikasi yang sering kita gunakan. Pada aktivitas ini, kita akan melihat bagaimana proses pengurutan dapat dilakukan dengan menggunakan berbagai strategi. Pelajarilah strateginya!

Pengurutan merupakan suatu permasalahan klasik pada komputasi yang dilakukan untuk mengatur agar suatu kelompok benda, objek, atau entitas diletakkan mengikuti aturan tertentu. Urutan yang paling sederhana misalnya mengurutkan angka secara terurut menaik atau menurun.

Biasanya, masalah pengurutan terdiri atas sekumpulan objek yang disusun secara acak yang harus diurutkan. Setelah itu, secara sistematis, posisi objek diperbaiki dengan melakukan pertukaran posisi dua buah objek. Hal ini dilakukan secara terus-menerus hingga semua posisi objek benar.

Misal, kita memperoleh 5 buah angka acak berikut:

Kita dapat membuat angka tersebut terurut menaik dengan melakukan satu kali pertukaran, yaitu dengan menukar nilai 4 dengan nilai 3. Terdapat 2 langkah penting dalam melakukan sebuah pengurutan. Langkah pertama ialah melakukan pembandingan. Untuk melakukan pengurutan, dipastikan ada dua buah nilai yang dibandingkan. Pembandingan ini akan menghasilkan bilangan yang lebih besar dari, lebih kecil dari, atau memiliki nilai sama dengan sebuah bilangan lainnya. Langkah kedua ialah melakukan penempatan bilangan setelah melakukan pembandingan. Penempatan bilangan ini dilakukan setelah didapatkan bilangan lebih besar atau lebih kecil (bergantung pada pengurutan yang digunakan).

Terdapat beberapa teknik (algoritma) untuk melakukan pengurutan seperti *bubble sort, insertion sort, quick sort, merge sort, dan selection sort*. Pada unit ini, hanya akan diberikan penjelasan untuk setiap tiga teknik ialah sebagai berikut. Teknik lainnya dapat kalian pelajari dari referensi yang diberikan.

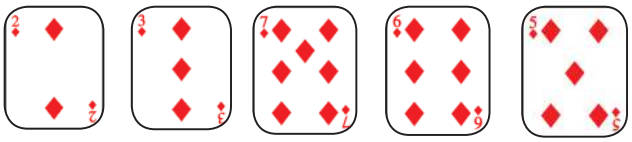
1. *Insertion Sort*

Insertion Sort adalah salah satu algoritma yang digunakan untuk permasalahan pengurutan dalam list (daftar objek). Sesuai namanya, *insertion sort* mengurutkan sebuah list dengan cara menyisipkan elemen satu per satu sesuai dengan urutan besar kecilnya elemen hingga semua elemen menjadi list yang terurut. Misalnya, dalam kasus mengurutkan elemen list dari yang terkecil hingga terbesar (*ascending*), tahap pertama ialah kita akan membaca suatu elemen dengan elemen yang berdekatan. Apabila elemen yang berdekatan dengan elemen saat ini lebih kecil, elemen yang lebih kecil akan ditukar

dengan elemen yang lebih besar dan dibandingkan kembali dengan elemen-elemen sebelumnya yang sudah terurut. Apabila elemen saat ini sudah lebih besar dari elemen sebelumnya, iterasi berhenti. Hal ini dijalankan satu per satu hingga semua list menjadi terurut.

Ilustrasi Insertion Sort

Terdapat sebuah deret bilangan seperti berikut: 2, 3, 7, 6, 5 yang direpresentasikan dengan menggunakan kartu. Urutkan bilangan tersebut secara menaik dengan menggunakan algoritma insertion sort.



Proses Iterasi Pertama

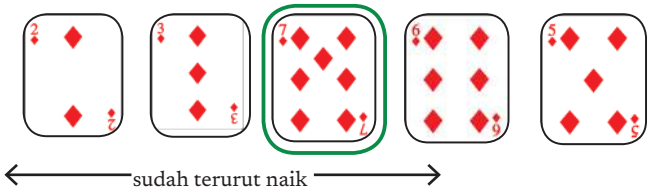
Langkah pertama, tinjau bilangan kedua, bandingkan bilangan pertama dan kedua, yaitu 2 dan 3. Didapatkan 2 lebih kecil dari 3, maka urutan bilangan tersebut tetap (2,3).

(2, **3**, 7, 6, 5) menjadi (2, **3**, 7, 6, 5)

Proses Iterasi Kedua

Pada iterasi selanjutnya, kita mengambil bilangan ketiga, yaitu 7. Lalu bandingkan dengan bilangan sebelumnya. Karena 3 lebih kecil dari 7, urutan tetap.

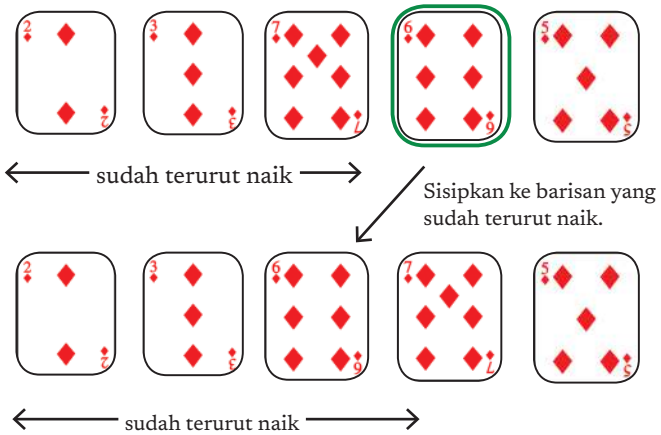
(2, 3, **7**, 6, 5) menjadi (2, 3, **7**, 6, 5)



Proses Iterasi Ketiga

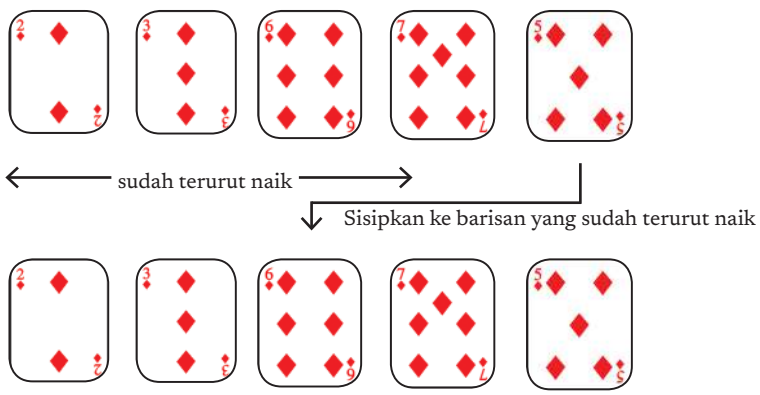
Pada iterasi selanjutnya, kita mengambil bilangan keempat, yaitu 6. Lalu, bandingkan dengan bilangan sebelumnya. Didapatkan bahwa 7 lebih besar dari 6. Oleh karena itu, selanjutnya, kita akan membandingkan dengan bilangan-bilangan sebelumnya, lalu menukarnya apabila bilangan tersebut lebih besar. Pertama, kita akan membandingkan 6 dan 7. Apakah 6 lebih kecil dari 7? Karena iya, kita akan menukar 6 dengan 7. Lalu, kita akan membandingkan lagi dengan bilangan sebelumnya, yaitu 3. Apakah 6 lebih kecil dari 3? Karena 6 tidak lebih kecil dari 3, maka 6 sudah berada pada posisi yang benar, yaitu sebelum 7 dan setelah 3.

Proses memindahkan 6 di antara 3 dan 7 ini biasa disebut penyisipan (insertion) sehingga nama algoritma ini disebut insertion sort.
(2, 3, **7**, **6**, 5) menjadi (2, 3, **6**, **7**, 5)



Proses Iterasi Keempat

Pada iterasi selanjutnya, kita mengambil bilangan kelima, yaitu 5. Didapatkan bahwa 7 lebih besar dari 5. Oleh karena itu, selanjutnya, kita akan membandingkan dengan bilangan-bilangan sebelumnya, lalu menukarnya apabila bilangan tersebut lebih besar. Pertama, kita akan membandingkan 5 dan 6. Apakah 5 lebih kecil dari 6? Karena iya, kita akan menukar 5 dengan 6. Setelah itu, kita akan mengecek dengan bilangan sebelumnya lagi, yaitu 3. Apakah 5 lebih kecil dari 3? Karena 5 tidak lebih kecil dari 3, maka 5 sudah pada posisi seharusnya, yaitu setelah 3 dan sebelum 6. Terjadi lagi proses penyisipan kartu 5 di antara 3 dan 6.
(2, 3, 6, 7, **5**) menjadi (2, 3, **5**, 6, 7)



3. *Selection sort*

Selection sort merupakan algoritma pengurutan yang juga cukup sederhana, dengan algoritma mencari (menyeleksi) bilangan terkecil/terbesar (bergantung pada urut naik atau turun) dari daftar bilangan yang belum terurut dan

meletakkannya dalam daftar bilangan baru yang dijaga keterurutannya. Algoritma ini membagi daftar bilangan menjadi dua bagian, yaitu bagian terurut dan bagian yang belum terurut. Bagian yang terurut di sebelah kiri dan bagian yang belum terurut di sebelah kanan. Awalnya, semua elemen bilangan dalam daftar ialah bagian yang belum terurut, dan bagian yang terurut kosong. Berikut langkah-langkah yang terdapat pada algoritma *selection sort*.

1. Cari bilangan terkecil yang ada pada bagian belum terurut.
2. Tukar bilangan tersebut dengan bilangan pertama bagian belum terurut, lalu masukkan ke bagian terurut.
3. Ulangi langkah 1 dan 2 sampai bagian yang belum terurut habis.

Ilustrasi urut-urutan *selection sort* dapat dilihat pada tabel berikut.

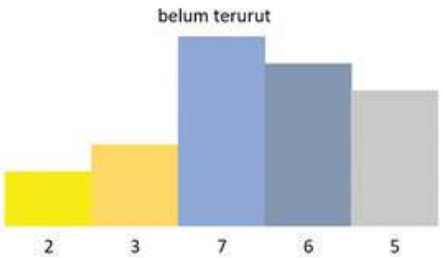
Bagian terurut	Bagian yang belum terurut	Nilai terkecil dari bagian belum terurut
()	(2,3,7,6,5)	2
(2)	(3,7,6,5)	3
(2,3)	(7,6,5)	5
(2,3,5)	(6,7)	6
(2,3,5,6)	(7)	7
(2,3,5,6,7)	()	

Secara rinci, algoritma *selection sort* yang dikaitkan dengan pemrograman dijelaskan sebagai berikut.

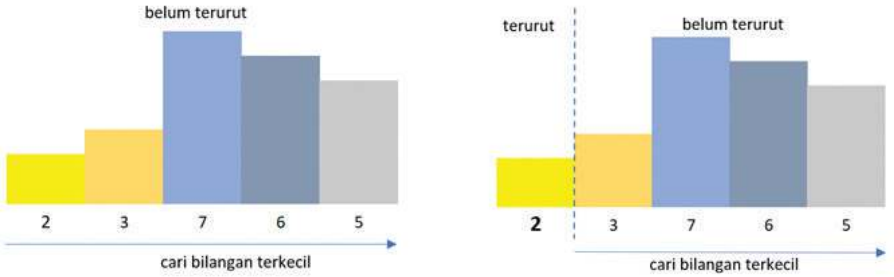
Terdapat sebuah daftar bilangan tidak terurut seperti berikut: 2, 3, 7, 6, 5. Urutkan bilangan tersebut secara menaik dengan menggunakan algoritma *selection sort.*

Proses Iterasi Pertama

Data Awal:



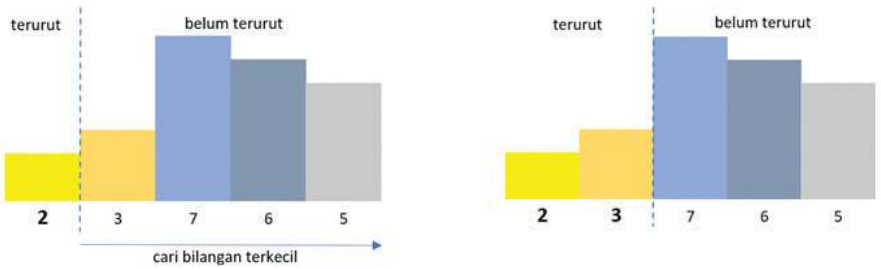
Cari bilangan terkecil di bagian belum terurut: ditemukan 2 sebagai bilangan terkecil.



Tukar bilangan 2 dengan bilangan pertama bagian belum terurut. Geser batas bagian yang sudah terurut ke kanan sehingga 2 menjadi bagian yang sudah terurut. Dalam ilustrasi ini, angka yang dicetak tebal menunjukkan bilangan yang sudah terurut.

Proses Iterasi Kedua

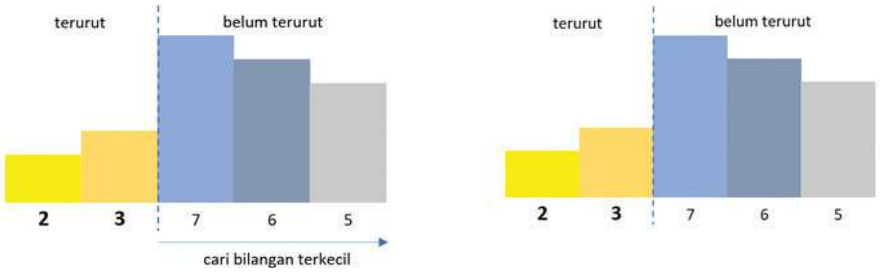
Cari bilangan terkecil di bagian belum terurut, ditemukan angka 3 sebagai bilangan terkecil.



Tukar bilangan 3 dengan bilangan pertama bagian belum terurut. Geser batas bagian yang sudah terurut ke kanan sehingga 3 menjadi bagian yang sudah terurut.

Proses Iterasi Ketiga

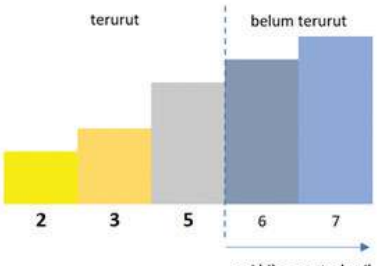
Cari bilangan terkecil di bagian belum terurut, ditemukan angka 5 sebagai bilangan terkecil.

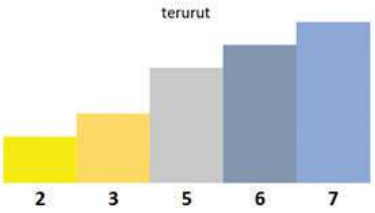


Tukar bilangan 5 dengan bilangan pertama bagian belum terurut, yaitu 7. Geser batas bagian yang sudah terurut ke kanan, sehingga 5 menjadi bagian yang sudah terurut.

Proses Iterasi Keempat

Cari bilangan terkecil di bagian belum terurut, ditemukan angka 6 sebagai bilangan terkecil.





Tukar bilangan 6 dengan bilangan pertama bagian belum terurut. Di bagian akhir, karena data tinggal dua, setelah proses penukaran, algoritma telah selesai dilaksanakan.

Aktivitas Individu/Berkelompok

Aktivitas BK-K10-02: Bermain Kartu

Apa yang kalian perlukan?

10 kartu yang masing-masing bertuliskan angka 1 sampai 10.

Skenario Permainan

Aktivitas dapat dilakukan secara mandiri atau berkelompok.

1. Kalian akan diberikan sebuah kartu bertuliskan angka dari 1 - 10.
2. Kelima belas kartu tersebut kalian kocok dan letakkan dalam bentuk barisan di atas meja. Kartu diletakkan tertutup.
3. Kalian harus dapat mengurutkan semua kartu secara menaik. Kartu yang berada di paling kiri barisan harus yang paling kecil.
4. Untuk mengurutkan, kalian harus melakukan serangkaian pertukaran kartu. Pertukaran dilakukan dengan membuka dua buah kartu. Apabila diperlukan, kalian dapat menukar posisi kedua kartu tersebut.
5. Kalian diminta untuk menyusun algoritma pertukaran yang dapat dilakukan untuk memastikan semua kartu dalam posisi terurut. Kalian dapat memilih untuk menggunakan salah satu dari tiga algoritma pengurutan yang disampaikan pada bagian konsep.

Ayo Berdiskusi

Apa yang kalian diskusikan?

Setelah bermain, saatnya memikirkan permainan tersebut dan cara kalian bermain. Beberapa poin yang penting untuk didiskusikan seperti berikut.

1. Apakah permainan tadi merupakan masalah pengurutan?
2. Strategi pengurutan seperti apa yang kalian lakukan untuk melakukan pengecekan dan pertukaran sesedikit mungkin?
3. Apakah strategi kalian berbeda dengan strategi yang dilakukan oleh teman kalian? Jika berbeda, apa perbedaannya?
4. Strategi paling bagus apa yang dapat kalian temukan untuk mengurutkan dengan banyaknya pertukaran paling sedikit?
5. Adakah kondisi yang membuat kalian melakukan banyak sekali pertukaran untuk mengurutkan kartu secara menaik?

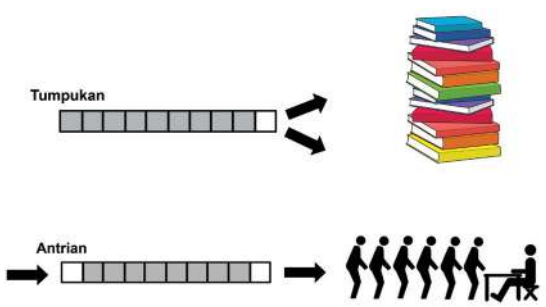
Ayo Kita Renungkan

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Apakah kalian sudah pernah melakukan permainan ini?
2. Saat mengurutkan kartu, apakah kalian senang?
3. Apakah kalian paham bahwa mengurutkan kartu itu suatu proses pengurutan?
4. Apakah kalian berhasil menemukan cara yang paling cepat untuk mengurutkan kartu tersebut?
5. Apakah kalian merasa ada masalah lain yang serupa dengan permainan tadi?
6. Pelajaran paling berkesan apa yang kalian dapatkan dari permainan ini?

C. Tumpukan (*Stack*) dan Antrean (*Queue*)

Kita akan mempelajari dua buah konsep cara penyimpanan data/objek dalam sebuah struktur yang akan menentukan urutan pemrosesan data/objek tersebut, yaitu tumpukan (stack) dan antrean (queue). Kedua konsep ini memiliki prosedur yang berbeda dalam menyimpan dan mengeluarkan data. Kedua konsep tersebut masing-masing memiliki peranan yang berbeda dan digunakan pada situasi yang berbeda pula.



Bayangkan sebuah loket di sebuah rumah sakit, di mana para pasien yang akan berobat diminta untuk mendaftar lebih dahulu di loket penerimaan serta mengisi formulir pendaftaran. Setelah formulir tersebut diisi, para pasien akan mengembalikan formulir ke loket dan menunggu dipanggil oleh petugas. Kebetulan, di pagi hari, dokter yang bertugas belum datang sehingga para pasien harus menunggu. Ketika sang dokter tiba, petugas loket akan memanggil para pasien satu per satu untuk mendapat layanan.

Perhatikan sekarang bagaimana urutan pasien itu dipanggil oleh petugas loket.

1. Misalkan, petugas loket menumpuk formulir-formulir tersebut di mana formulir yang baru diterima diletakkan *di atas* formulir yang sudah diterima sebelumnya, kemudian ketika ketika memanggil pasien, petugas tersebut memanggil dengan urutan mulai dari formulir yang berada *di atas* tumpukan. Menurut kalian, apakah urutan tersebut adil/sesuai dengan yang diharapkan para pasien? Mengapa?
2. Bagaimana cara petugas menyusun tumpukan formulir dan/atau cara urutan memanggil para pasien dari tumpukan formulir sedemikian rupa sehingga pasien yang datang dan mengisi formulir lebih dulu, akan *dipanggil lebih dulu* juga (dan sebaliknya)?

Dalam dunia komputasi/informatika, terkadang, kita perlu untuk menyimpan data/objek dalam suatu urutan tertentu, untuk kemudian/sewaktu-waktu diambil/dikeluarkan kembali, mungkin untuk diproses lebih lanjut atau untuk tujuan-tujuan lain. Ada dua cara utama kita dapat melakukan penyimpanan ini.

1. Antrean (*queue*): pada metode ini, objek-objek disimpan dalam metode penyimpanan yang berupa sebuah antrean sehingga objek yang pertama/lebih dulu datang, juga akan lebih dulu keluar/selesai, layaknya sebuah antrean di loket, pintu masuk, dll. Prinsip ini disebut prinsip *First In First Out* (FIFO). Dalam sebuah antrean orang, misalnya, jelas orang yang pertama datang akan berada di depan antrean, dan harus menjadi yang pertama yang mendapat pelayanan.
2. Tumpukan (*stack*): pada metode ini, objek-objek disimpan dalam metode penyimpanan yang menyerupai sebuah tumpukan (misal: tumpukan piring). Dengan demikian, objek yang *pertama/lebih dulu* disimpan justru akan menjadi yang terakhir keluar. Prinsip ini disebut juga *Last In First Out* (LIFO). Dalam tumpukan piring, misalnya, piring pertama yang diletakkan akan berada di posisi paling bawah, dan jika kita ambil piring satu per satu dari tumpukan itu, tentunya piring yang berada di posisi paling bawah tersebut akan menjadi yang terakhir diambil.

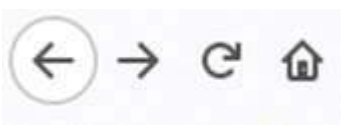
Baik dalam kehidupan sehari-hari maupun dalam dunia informatika, kedua konsep urutan penyimpanan data tersebut memiliki peran dan kegunaan masing-masing. Ada permasalahan-permasalahan/situasi di mana antrean (FIFO) lebih cocok digunakan. Sebaliknya, ada juga permasalahan-permasalahan di mana tumpukan (LIFO) lebih tepat diterapkan. Untuk lebih memahami kedua konsep ini dan bagaimana mereka digunakan, mari, kita lakukan beberapa aktivitas di bawah ini.

Aktivitas Berkelompok

Aktivitas BK-K10-03-U: Penggunaan *Stack* dan *Queue* secara Tepat (*Unplugged*)

Pada aktivitas ini, kalian akan membaca beberapa skenario kondisi, baik dalam dunia sehari-hari maupun dalam dunia informatika. Tugas kalian ialah memikirkan, pada setiap kondisi/skenario tersebut, manakah yang lebih tepat digunakan/lebih relevan menggambarkan situasi tersebut, apakah stack ataukah queue. Berikan penjelasan mengapa kalian memilih jawaban tersebut!

1. Di persimpangan jalan, terdapat lampu merah. Apabila lampu merah menyala, mobil-mobil yang datang ke persimpangan tersebut harus berhenti dulu. Ketika lampu berubah menjadi hijau, semua mobil perlahan-lahan berjalan kembali dalam urutan tertentu. Manakah yang lebih tepat menggambarkan situasi tersebut?
2. Ketika menjelajah web/internet, kita menggunakan sebuah *browser* (misal Firefox, Chrome dll). Terdapat sebuah fitur yang memungkinkan kita untuk bergerak dari satu halaman yang sudah kita kunjungi ke halaman lainnya, yaitu dengan menekan tombol *Back* dan *Forward*. Misalnya, kita mengunjungi halaman A, kemudian B, lalu C. Jika kita kemudian menekan tombol *Back*, dari halaman C kita akan kembali ke halaman B. Jika kita tekan lagi tombol Back (pada saat ada di B), kita akan kembali ke A. Jika kemudian kita tekan tombol *Forward*, kita akan kembali halaman B, dan jika kita tekan sekali lagi tombol Forward, kita akan kembali ke halaman C. Oleh karena itu, aplikasi browser tersebut harus menyimpan (dan mengingat) semua halaman yang sudah pernah kita kunjungi sebelumnya (biasa disebut Riwayat atau *History*). Bentuk penyimpanan yang manakah (*stack* atau *queue*) yang paling tepat digunakan untuk menyimpan Riwayat pada *browser*?



Gambar 2.3 Tombol Back dan Forward pada Firefox
Sumber: Dokumen Kemendikbud, 2021

3. Mesin *printer* bertugas untuk mencetak dokumen yang dikirimkan dari sebuah komputer. Satu buah printer dapat terhubung ke beberapa buah komputer sekaligus, dan semuanya dapat mengirim perintah kepada *printer* tersebut untuk mencetak dokumen yang berbeda-beda. *Printer* tersebut tentunya hanya bisa mencetak satu buah dokumen dalam satu waktu tertentu, dan mungkin membutuhkan beberapa detik/menit untuk menyelesaikan proses cetak satu dokumen. Oleh karena itu, ketika *printer* sedang sibuk mencetak sebuah dokumen dari sebuah komputer, kemudian datang permintaan mencetak dari beberapa komputer yang lain (yang berbeda). *Printer* tersebut harus menyimpan dokumen-dokumen yang baru datang tersebut agar nanti dapat dicetak ketika proses pencetakan yang sedang berjalan saat ini sudah selesai. Manakah yang lebih tepat digunakan, *stack* atau *queue* untuk penyimpanan dokumen-dokumen yang sedang "menunggu giliran" untuk dicetak tadi?
4. Pada sebuah aplikasi pengolah dokumen, biasanya terdapat fasilitas untuk melakukan *Undo* dan *Redo*. Operasi *Undo* akan membatalkan langkah/tindakan terakhir yang kita lakukan saat mengedit dokumen (misal, jika kita menyadari ada kesalahan pada langkah terakhir kita), sedangkan *Redo* digunakan untuk mengulang kembali operasi yang baru saja dibatalkan dengan sebuah *Undo*. Proses *Undo* dan *Redo* ini dapat dilakukan sampai dengan operasi pertama setelah sebuah dokumen dibuka/disimpan. Misalnya, terjadi rangkaian kejadian berikut:
   a. Budi membuka dokumen A
   b. Budi menambahkan judul pada dokumen A
   c. Budi menulis sebuah paragraf pada dokumen A
   d. Budi menambahkan sebuah tabel pada dokumen A
   e. Budi menyisipkan sebuah gambar pada dokumen A

   Apabila kemudian Budi menekan tombol *Undo*, operasi terakhir (yaitu penambahan gambar) akan dibatalkan sehingga gambar tersebut akan hilang dari dokumen. Jika kemudian Budi menekan tombol *Undo* sekali lagi, operasi terakhir sebelum itu (yaitu menambahkan tabel) juga akan dibatalkan sehingga tabel tersebut akan hilang dari dokumen. Jika kemudian Budi menekan tombol *Redo*, operasi *Undo* yang terakhir (yaitu yang menghilangkan tabel) akan dibatalkan sehingga tabel tersebut akan muncul kembali.

Jelas bahwa aplikasi perlu untuk menyimpan data-data berupa tindakan/operasi apa saja yang dilakukan oleh penggunanya dari awal sampai akhir, serta efeknya terhadap dokumen agar dapat memberikan fungsionalitas *Undo* dan *Redo* tersebut. Manakah di antara *stack* dan *queue* yang lebih tepat digunakan untuk menyimpan operasi-operasi tersebut?

Lembar Kerja Siswa

Untuk setiap kasus di atas, lakukan analisis penggunaan stack dan queue dengan mengisi LKS ini.

Persoalan	Stack	Queue	Saya Pilih ...
Persimpangan lampu merah			
Penjelajahan internet			
Antrean permintaan print dokumen dalam sebuah komputer			
Undo Redo			

Ayo Lakukan

Aktivitas Berpasangan

Aktivitas BK-K10-04-U: Simulasi Stack

Pada aktivitas ini, kalian akan bermain dengan satu orang siswa lainnya. Satu orang harus berperan menjadi *Pemberi Perintah* dan satu lagi harus berperan sebagai *Simulator*. Permainan dimulai dengan Pemberi Perintah memberikan sebuah perintah simulasi (yang akan dijelaskan di bawah). Kemudian, Simulator harus menjalankan simulasi dan memberikan jawaban yang benar. Jawaban tersebut harus diperiksa oleh Pemberi Perintah dan kemudian harus dinyatakan jawaban tersebut benar atau salah. Setelah itu, kedua orang bertukar peran: Simulator harus menjadi Pemberi Perintah dan Pemberi Perintah menjadi Simulator. Lakukan pertukaran ini sampai beberapa kali. Orang yang berhasil mendapatkan jawaban benar sebanyak mungkin akan menjadi pemenangnya.

Berikut ini format/bentuk perintah serta bentuk jawaban yang diinginkan.

Kita asumsikan ada sebuah *stack* yang mampu menyimpan nilai-nilai bilangan. Setiap perintah simulasi berisi kumpulan dari 2 buah perintah:

1. Push X
2. Pop

Perintah Push digunakan untuk menyimpan nilai ke dalam *stack*. Perintah ini harus diikuti oleh sebuah bilangan bulat X yang akan disimpan ke dalam *stack*. Perintah Pop digunakan untuk mengeluarkan angka yang *berada di atas tumpukan* saat ini. Jika saat ini tumpukan kosong, perintah Pop tidak memberikan efek apa-apa. Berikut ini contoh sebuah perintah simulasi dan hasilnya:

Perintah	Isi Stack	Hasil Pop
Push 5	*5*	
Push 3	*5,3*	

Perintah	Isi Stack	Hasil Pop
Push 2	5,3,2	
Push 4	5,3,2,4	
Pop	5,3,2	4
Push 6	5,3,2,6	
Pop	5,3,2	6
Pop	5,3	2
Pop	5	3

Ketika seorang Simulator menerima sebuah perintah simulasi, ia harus memberikan jawaban berupa daftar *bilangan yang akan dikeluarkan* dari *stack*, sesuai dengan urutan perintah simulasi yang ia terima. Misalnya, pada contoh di atas, Simulator harus memberikan jawaban berupa:

4
6
2
3

Tentunya, banyaknya angka pada jawaban harus sama dengan banyaknya perintah Pop yang diberikan oleh simulator.

Lembar Kerja Siswa

Untuk permainan peran ini dapat dipakai LKS berikut ini.

Pemberi Perintah	Catatan Simulator	Isi Stack	Hasil Pop

Jawaban Simulator:

Aktivitas BK-K10-05-U: Simulasi *Queue*

Pada aktivitas ini, kalian akan melakukan simulasi operasi pada sebuah *queue*. Serupa dengan aktivitas sebelumnya, aktivitas ini dijalankan oleh dua orang yang akan bertugas sebagai Pemberi Perintah dan Simulator. Pemberi Perintah akan memberikan kumpulan perintah yang berisi operasi pada *queue*, sedangkan Simulator harus memberikan jawaban berupa rangkaian isi *queue* yang dihasilkan dari setiap perintah yang diberikan.

Format perintah ialah sebagai berikut.

1. Enqueue X: memasukkan sebuah bilangan bulat ke dalam *queue.*
2. Dequeue: membuang/mengeluarkan bilangan yang berada pada posisi pertama antrean.

Untuk setiap perintah, Simulator harus menuliskan *apa isi queue* setiap kali perintah tersebut selesai dijalankan. Sebagai contoh, Pemberi Perintah memberikan perintah-perintah sebagai berikut.

Perintah	Simulator menulis isi queue setelah setiap perintah dijalankan	Hasil Dequeue
Enqueue 5	5	
Enqueue 3	5, 3	
Dequeue	3	5
Enqueue 4	3, 4	
Dequeue	4	3

Jika Simulator harus memberikan 5 baris jawaban berupa isi dari *queue* setelah setiap perintah dijalankan, hasilnya:

1. 5
2. 5, 3
3. 3
4. 3, 4
5. 4

Lembar Kerja Siswa

Untuk permainan peran ini dapat dipakai LKS.

Pemberi Perintah	Catatan Simulator	Isi *Stack*	Hasil Pop

Jawaban Simulator:

Jawablah pertanyaan berikut dalam Lembar Refleksi pada Buku Kerja. Jangan lupa mencatat kegiatan dalam Jurnal.

1. Apakah kalian dapat memahami dengan baik perbedaan dari konsep *stack* dan *queue*?
2. Jika diberikan sebuah kondisi di dunia nyata/informatika, dapatkah kalian menentukan apakah stack atau queue yang lebih relevan diterapkan sebagai metode penyimpanan?

3. Dapatkah kalian mencari contoh-contoh lain penerapan stack dan queue dalam kehidupan sehari-hari?
4. Apakah kalian dapat memainkan permainan simulasi stack dan queue di atas dengan baik? Apakah permainan tersebut membantu proses pemahaman kalian terhadap kedua konsep tersebut??

Ingin Tahu Lebih?

Jika ingin belajar lebih mendalam tentang materi di atas, kalian bisa mengunjungi tautan berikut ini.

Pencarian (*Searching*)

1. Search Algorithm: https://en.wikipedia.org/wiki/Search_algorithm
2. Binary Search: https://khanacademy.org/computing/computer-science/algorithms/binary-search

Pengurutan (*Sorting*)

1. Sorting Algorithm: https://en.wikipedia.org/wiki/Sorting_algorithm
2. Video Bubble Sort:https://youtu.be/nmhjrI-aW5o
3. Video Insertion Sort: https://youtu.be/OGzPmgsI-pQ
4. Video Selection Sort:https://youtu.be/xWBP4lzkoyM

Rekursi

1. Rekursi: https://en.wikipedia.org/wiki/Recursion

Graf:

1. Graf: https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)
2. Penelusuran graf: https://en.wikipedia.org/wiki/Graph_traversal