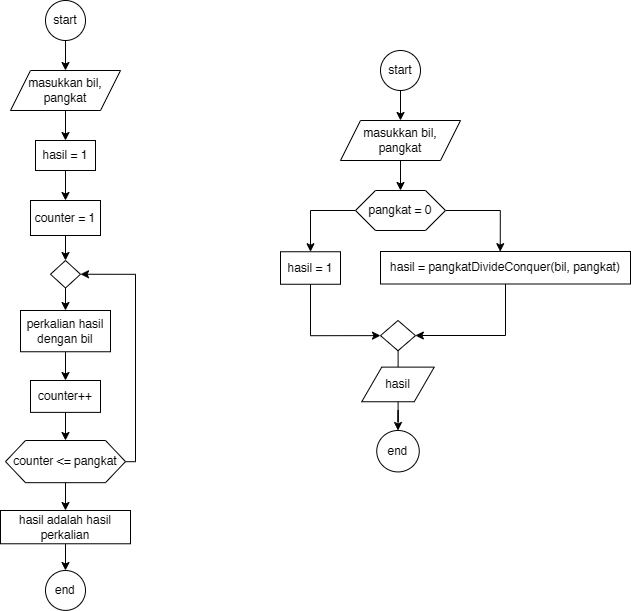
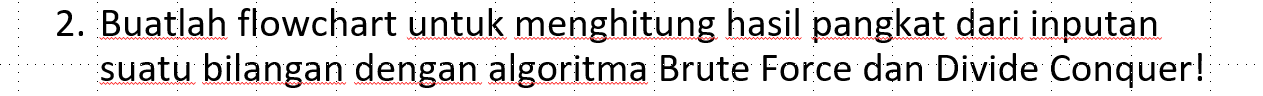
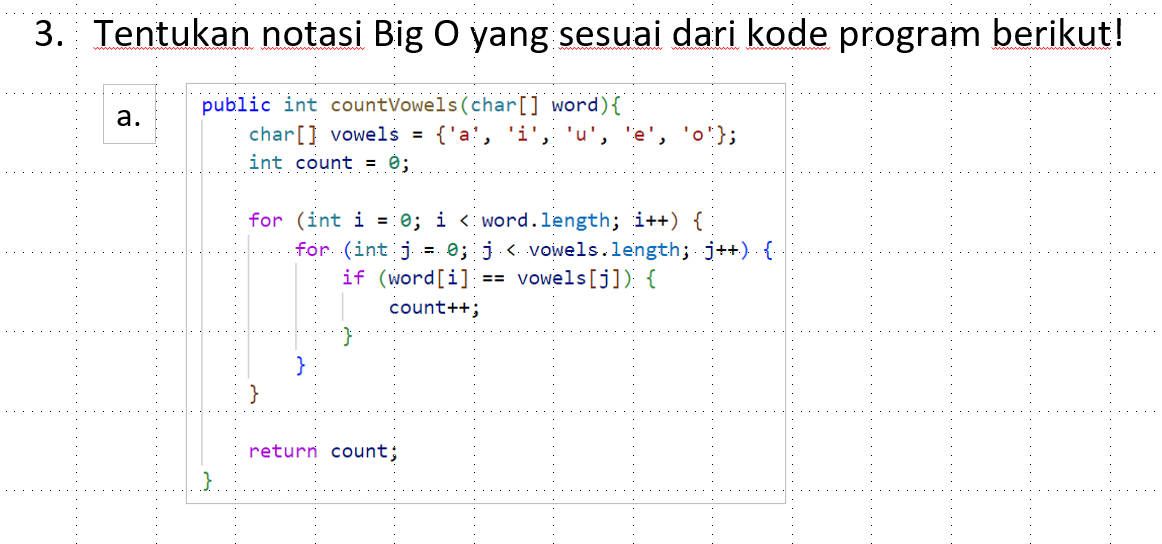
Nama : Muhammad Abhinaya Zurfa  
Kelas : SIB 1 C / 20



notasi big o pada kode tersebut adalah O(n\*m)   
dimana n mewakili Panjang word.length dan m mewakili Panjang vowels.length

Kode program yang Anda berikan adalah sebuah fungsi untuk menghitung jumlah huruf vokal dalam sebuah array karakter. Fungsi ini menggunakan dua buah loop bersarang: loop luar iterasi melalui setiap karakter dalam array word, dan loop dalam iterasi melalui array vowels untuk memeriksa apakah karakter tersebut adalah vokal.

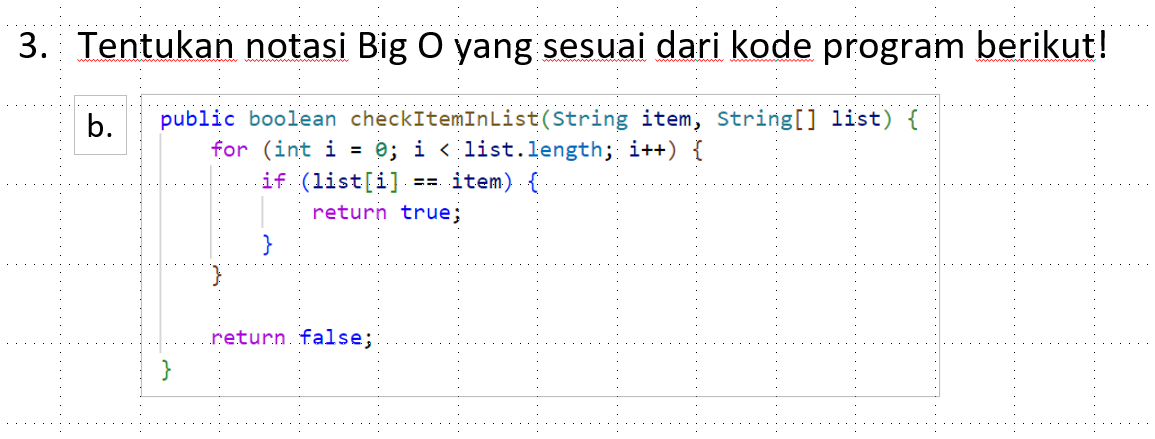
Notasi Big O untuk fungsi ini adalah **O(n \* m)**, di mana:

* **n** adalah panjang dari array word
* **m** adalah panjang dari array vowels

Alasan notasi Big O adalah **O(n \* m)**:

* Loop luar berjalan sebanyak **n** kali, yang sesuai dengan jumlah karakter dalam array word.
* Untuk setiap iterasi loop luar, loop dalam berjalan sebanyak **m** kali, yang sesuai dengan jumlah vokal yang diperiksa.
* Karena loop dalam bersarang di dalam loop luar, total jumlah operasi yang dilakukan adalah hasil kali dari jumlah iterasi kedua loop tersebut, yaitu **n \* m**.

Dengan demikian, waktu eksekusi dari fungsi ini akan meningkat secara linier dengan panjang dari array word dan jumlah vokal yang diperiksa, sehingga kompleksitas waktunya adalah linier terhadap ukuran input, yang ditunjukkan dengan notasi Big O **O(n \* m)**. Ini menunjukkan bahwa fungsi ini tidak efisien untuk input yang sangat besar karena jumlah operasi yang dibutuhkan meningkat secara signifikan dengan ukuran input.



notasi big o pada kode tersebut adalah O(n)  
dimana n mewakili Panjang dari string list.length dan pada kode tersebut hanya menggunakan 1 looping

Fungsi checkItemInList yang Anda berikan adalah untuk memeriksa apakah suatu item terdapat dalam sebuah list. Fungsi ini menggunakan loop tunggal yang iterasi melalui setiap elemen dalam array list dan membandingkannya dengan item.

Notasi Big O untuk fungsi ini adalah **O(n)**, di mana **n** adalah panjang dari array list.

Alasan notasi Big O adalah **O(n)**:

* Loop berjalan sebanyak **n** kali, yang sesuai dengan jumlah elemen dalam array list.
* Pada kasus terbaik, jika item yang dicari adalah elemen pertama dalam list, fungsi akan segera mengembalikan true dan kompleksitas waktunya adalah **O(1)**.
* Pada kasus terburuk, jika item tidak ada dalam list atau berada di posisi terakhir, loop akan berjalan sebanyak **n** kali, sehingga kompleksitas waktunya adalah **O(n)**.
* Karena kita menilai kompleksitas algoritma berdasarkan kasus terburuk, maka notasi Big O dari fungsi ini adalah **O(n)**.

Ini menunjukkan bahwa waktu eksekusi dari fungsi ini akan meningkat secara linier dengan panjang dari array list. Fungsi ini efisien untuk list dengan ukuran kecil hingga sedang, tetapi untuk list yang sangat besar, jumlah operasi yang dibutuhkan bisa menjadi signifikan.