Laporan Praktikum Algoritma & Pemrograman

Semester Genap 2025/2026

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

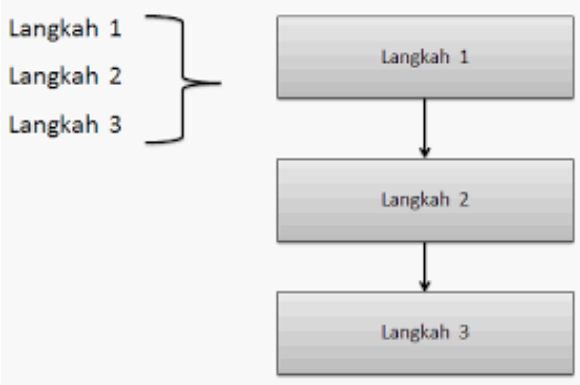
NIM	**71251218**
Nama Lengkap	**Agatha Sabda Alethea Prabawa**
Minggu ke / Materi	**03 / Flowchart dan Pseudocode**

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2026

BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

Algoritma



algoritma

Algoritma adalah serangkaian instruksi atau langkah-langkah yang logis, sistematis, dan terurut yang dirancang untuk menyelesaikan suatu masalah komputasi tertentu (Prasatya, 2025). Algoritma disusun secara berurutan sehingga komputer dapat mengeksekusinya secara efisien. Beberapa fungsi dari algoritma sebagai berikut (Dicoding, 2025):

1. Menyederhanakan proses pemecahan masalah.
2. Meningkatkan efisiensi program.
3. Mendukung perencanaan dan pengujian.
4. Meningkatkan skalabiilitas.

Algoritma memiliki 3 macam notasi, yaitu:

1. Uraian deskriptif
2. *Flowchart*/Diagram alir
3. *Psudocode*

Uraian Deskriptif

Seperti yang tertulis diatas, algoritma memiliki fungsi untuk menyederhanakan proses pemecahan masalah dan mendukung perencanaan. Jika kalian pernah membaca tata cara atau urutan membuat suatu makanan, tata cara tersebut merupakan contoh algoritma pemecahan masalah dalam kehidupan sehari-hari. Sebagai contoh, tata cara memasak mie instan:

1. Masak 350 ml air dalam panci hingga mendidih/matang.
2. Buka bungkus mie instan, lalu masukan mie kedalam panci dan tunggu selama 3 menit.
3. Siapkan mangkok, lalu keluarkan bumbu dari bungkus dan masukan ke dalam mangkok.
4. Tiriskan mie dan tuangkan ke dalam mangkok, lalu campur mie dengan bumbu sampai merata.
5. Mie instan siap disajikan.

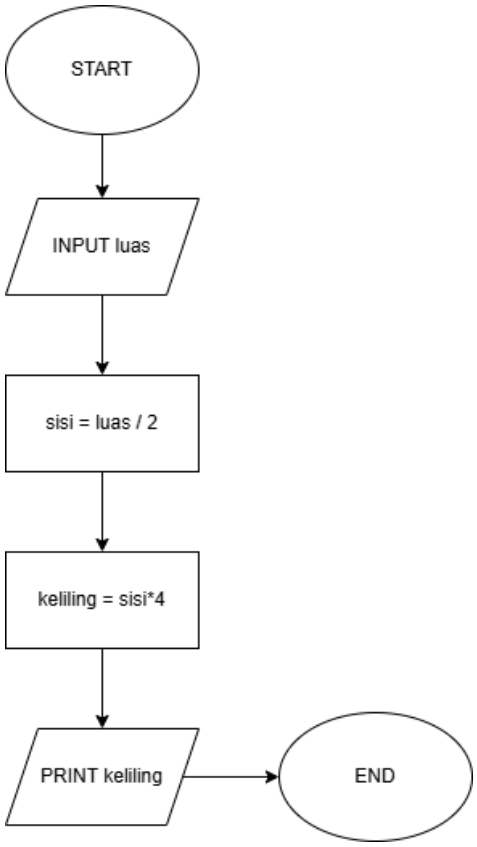
Tata cara pembuatan mie instan ini sudah ditulis sesuai dengan urutan. Jika kita mengacak urutan misal dengan melakukan langkah 2 lalu baru melakukan langkah 1, maka hasilnya tidak akan sesuai harapan atau tidak akan maksimal. Hal ini juga akan terjadi di pemrograman, jika kita tidak melakukan proses sesuai dengan algoritma pemecahan masalah yang telah dibuat, maka masalah tersebut tidak akan terselesaikan dengan baik.

Flowchart / Diagram Alir

Flowchart merupakan sebuah representasi visual yang mengilustrasikan langkah dan logika dari program komputer menggunakan simbol dan konektor. *Flowchart* biasa juga disebut dengan diagram alir atau bagan alur. *Flowchart* memiliki beberapa manfaat sebagai berikut,

1. Representasi algoritma dalam bentuk visual. Hal ini membuat alur algorimta menjadi lebih mudah dipahami, urutan algoritma telihat jelas, dan *flowchart* memiliki bentuk yang standar.
2. Sebagai salah satu alat komunikasi orang teknis dan orang non-teknis dalam satu tim pengembang aplikasi.
3. Membantu memperjelas proses yanng kompleks sehingga lebih mudah dipahami.

Dalam pemrograman, *flowchart* digunakan untuk membantu memecahkan masalah-masalah dalam menjalankan suatu operasi program. Maka dari itu, *flowchart* harus dapat merepresentasikan segala elemen dalam bahasa pemrograman. **Contoh** *flowchart* dalam operasi menghitung luas keliling persegi yang luasnya sudah diketahui:



Algoritma:

1. Masukan nilai luas (luas).
2. Hitung panjang sisi (sisi = luas/2)
3. Hitung keliling (keliling = sisi x 4)
4. Tampilkan hasil keliling (PRINT keliling)

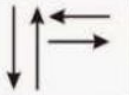
n/p:

- Input luas (luas) = menyimpan nilai input luas.
- Panjang sisi (sisi) = menyimpan hasil perhitungan mencari nilai sisi.

- Panjang keliliing (keliling) = menyimpan hasil perhitungan mencari nilai keliling.

Notasi Flowchart

Untuk membuat *flowchart*, kita perlu mengetahui simbol-simbol/notasi-notasi apa yang harus digunakan untuk menjalankan suatu program dalam bentuk visual. *Flowchart* memiliki banyak simbol/notasi yang memiliki kegunaan atau arti yang berbeda-beda. Berikut daftar simbol/notasi dalam *flowchart* menurut ANSI (*American National Standart Institute*) (Indahyanti & Rahmawati, 2020):

	**Flow Direction symbol** Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.		**Simbol Manual Input** Simbol untuk pemasukan data secara manual on-line keyboard
	**Terminator Symbol** Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan		**Simbol Preparation** Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	**Connector Symbol** Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.		**Simbol Predefine Proses** Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	**Connector Symbol** Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.		**Simbol Display** Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	**Processing Symbol** Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer		**Simbol disk and On-line Storage** Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	**Simbol Manual Operation** Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer		**Simbol magnetik tape Unit** Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	**Simbol Decision** Simbol pemilihan proses berdasarkan kondisi yang ada.		**Simbol Punch Card** Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	**Simbol Input-Output** Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya		**Simbol Dokumen** Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

simbol-simbol dalam flowchart

Pada tabel tersebut, beberapa simbol/notasi yang paling sering digunakan yaitu:

1. *Flow Direction symbol* atau *connecting line*, digunakan untuk menghubungkan simbol satu dengan simbol yang lainnya.
2. *Terminator symbol,* digunakan untuk memulai dan mengakhiri algoritma (*start* dan *end*).

3. *Processing symbol*, digunakan untuk menunjukan proses pengolahan data yang dilakukan oleh komputer.
4. *Decision symbol*, digunakan untuk melakukan kondisi percabangan ketika terjadi proses pemilihan kondisi (*true/false* atau *yes/no*).
5. *Input-Output symbol*, digunakan untuk menyatakan proses *input* atau *output.*

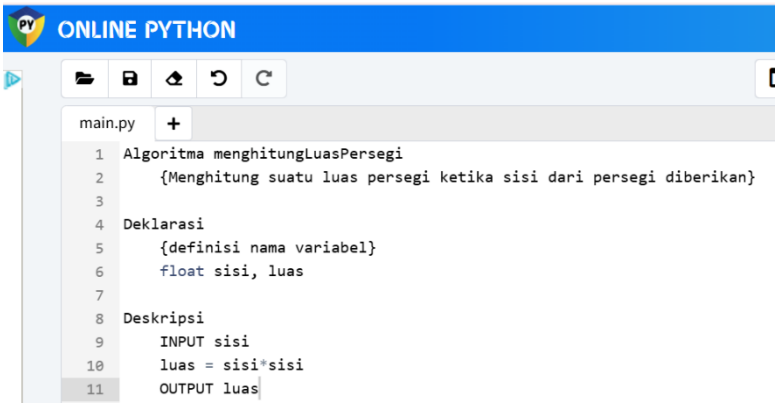
Pseudocode

Pseudocode adalah algoritma yang ditulis menyerupai kode program, tapi ditulis dalam bentuk yang lebih sederhana sehingga mudah untuk dipahami oleh manusia. *Pseudocode* memiliki beberapa fungsi, yaitu:

1. Sebagai representasi dari kode program tapi menggunakan bahasa yang lebih mudah dipahami oleh orang non-teknis.
2. Sebagai dokumentasi awal perancangan program untuk mambantu programmer menjelaskan alur awal program.
3. Mempermudah dalam menulis kode program karena dapat diterjemahkan ke dalam berbagai bahasa pemrograman.

Pseudocode dibagi menjadi 3 bagian:

1. Bagian kepala (Header).
2. Bagian deklarasi (definisi variabel).
3. Bagian deskripsi (rincian langkah).



contoh

Notasi Pseudocode

Pseudocode menerapkan beberapa kata kunci atau notasi yang umum dipahami. Beberapa notasi yang biasa atau sering digunakan pada *Pseudocode* adalah sebagai berikut:

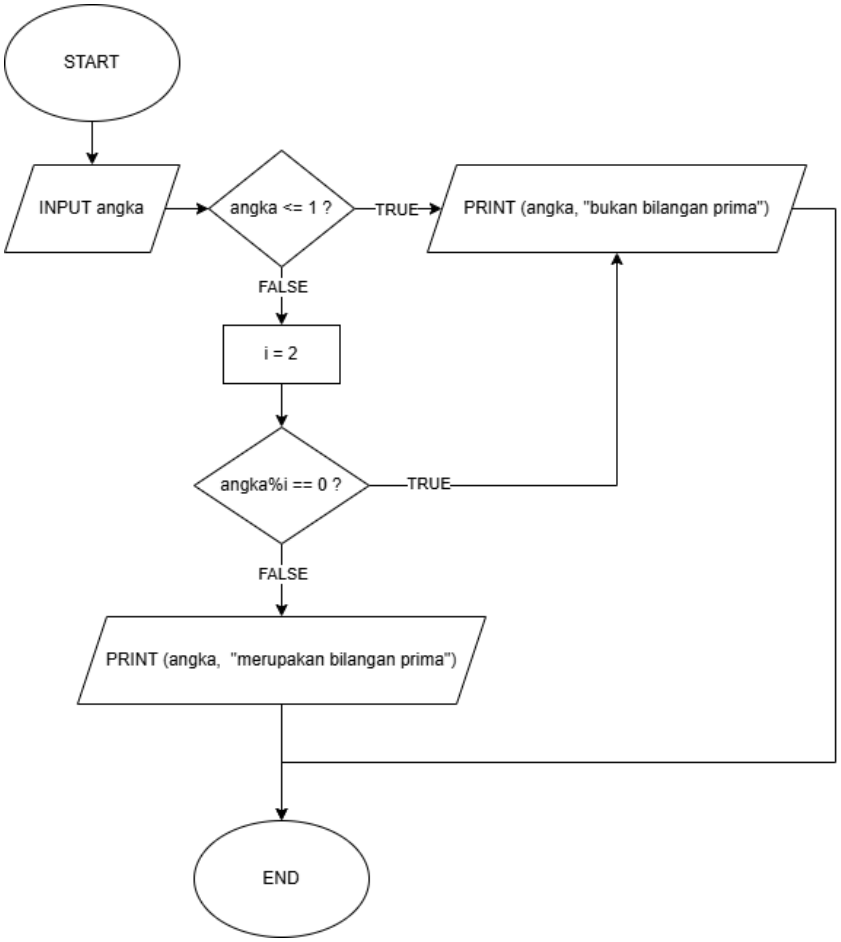
1. **INPUT**, digunakan untuk menunjukan proses penerimaan data atau nilai.
2. **OUTPUT**, digunakan untuk menunjukan hasil dari proses algoritma.
3. **FOR**, untuk melakukan perulangan dengan jumlah iterasi tertentu.
4. **WHILE**, untuk melakukan perulangan ketika jumlah iterasi belum diketahui dari awal.
5. **REPEAT-UNTIL**, untuk melakukan perulangan yang memiliki kondisi akhir.
6. **IF-THEN-ELSE**, menunjukan pengambilan keputusan berdasarkan kondisi logika.

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

Link github: https://github.com/gathasja/71251218_Agatha.git

SOAL 1



Alur ***flowchart***:

1. Algoritma dimulai (START).
2. User diminta untuk memberikan nilai angka (INPUT).
3. Pada percabangan, program akan mengecek apakah nilai angka yang diberi kurang dari sama dengan (<=) 1? Jika TRUE, maka angka tersebut bukan bilangan prima dan proses berakhir. Jika FALSE, maka proses berlanjut untuk pengecekan lebih lanjut.

4. Inisialisasi variabel i diatur menjadi 2.
5. Pada percabangan selanjutnya, program mengecek apakah modulo (sisa hasil bagi) dari nilai angka dengan i menghasilkan 0 (angka%i == 0)? Jika TRUE, maka angka tersebut bukan bilangan prima dan proses berakhir. Jika FALSE, maka angka tersebut merupakan bilangan prima.
6. Algoritma selesai (END).

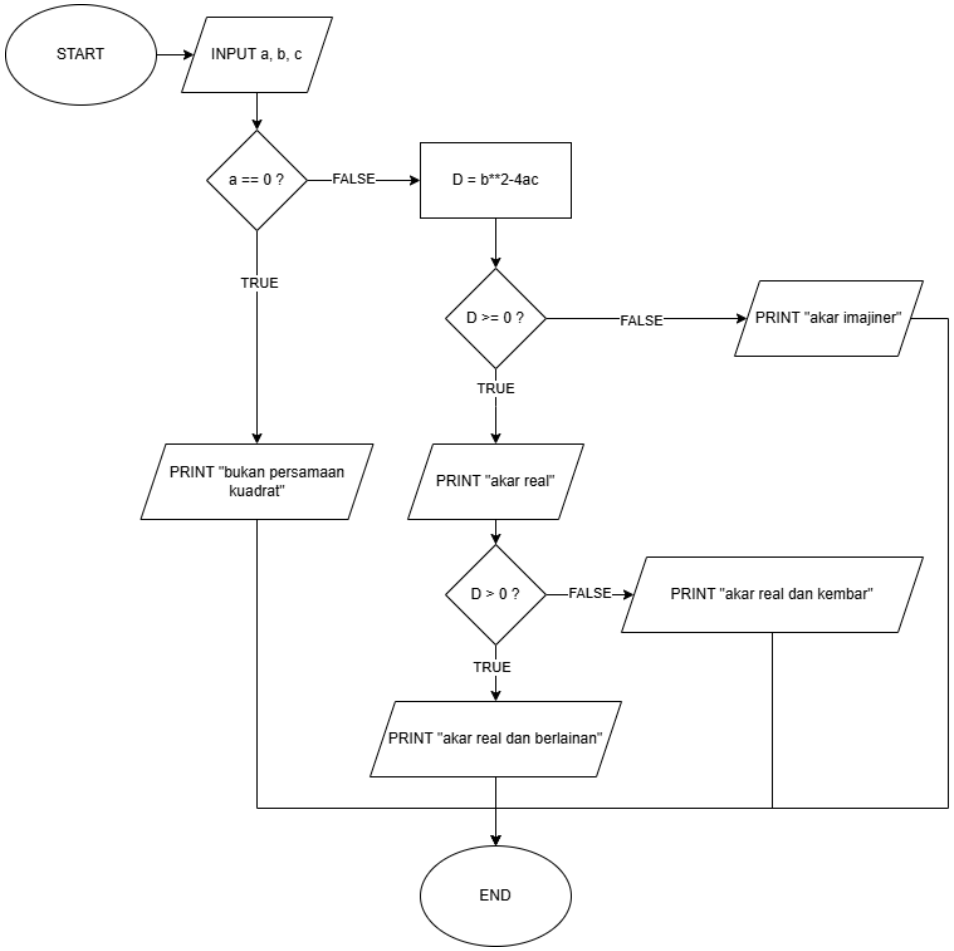
```
PROGRAM cekBilanganPrima
{program untuk menentukan apakah sebuah angka merupakan bilangan prima atau bukan}

DEKLARASI
   int angka, i

DESKRIPSI
    INPUT angka
    IF angka <= 1 THEN
        PRINT (angka, " bukan bilangan prima")
    ELSE
        i = 2
        IF angka % i == 0 THEN
            PRINT (angka, " bukan bilangan prima")
        ELSE
            PRINT (angka, " merupakan bilangan prima")
        ENDIF
    ENDIF
END
```

Alur ***pseudocode***:

1. Membuat sebuah header diawal algoritma.
2. Mendeklarasikan nama variabel dan tipe datanya (angka : integer, i : integer).
3. Mulainya algoritma:
    a. INPUT angka : user diminta untuk memasukan suatu nilai bilangan bulat yang akan disimpan pada variabel angka.
    b. Pemeriksaan pertama (IF angka <= 1 THEN): program memeriksa apakah angka kurang dari sama dengan 1. Jika TRUE, maka program akan menampilkan bahwa angka tersebut bukan bilangan prima.

c. Pemeriksaan kedua (modulo): jika angka tidak kurang dari sama dengan 1, maka program menginisialisasi variabel i diatur menjadi 2 (i = 2) dan program akan menghitung modulo (sisa hasil bagi) nilai angka dengan i untuk melihat apakah hasilnya 0. Jika TRUE, maka program akan menampilkan bahwa angka tersebut bukan bilangan prima. Jika FALSE, maka program akan menampilkan bahwa angka tersebut merupakan bilangan prima.
d. Program akan berakhir setelah salah satu kondisi terpenuhi dan hasil (OUTPUT) ditampilkan (ENDIF, ENDIF, END).

SOAL 2



Alur ***flowchart***:

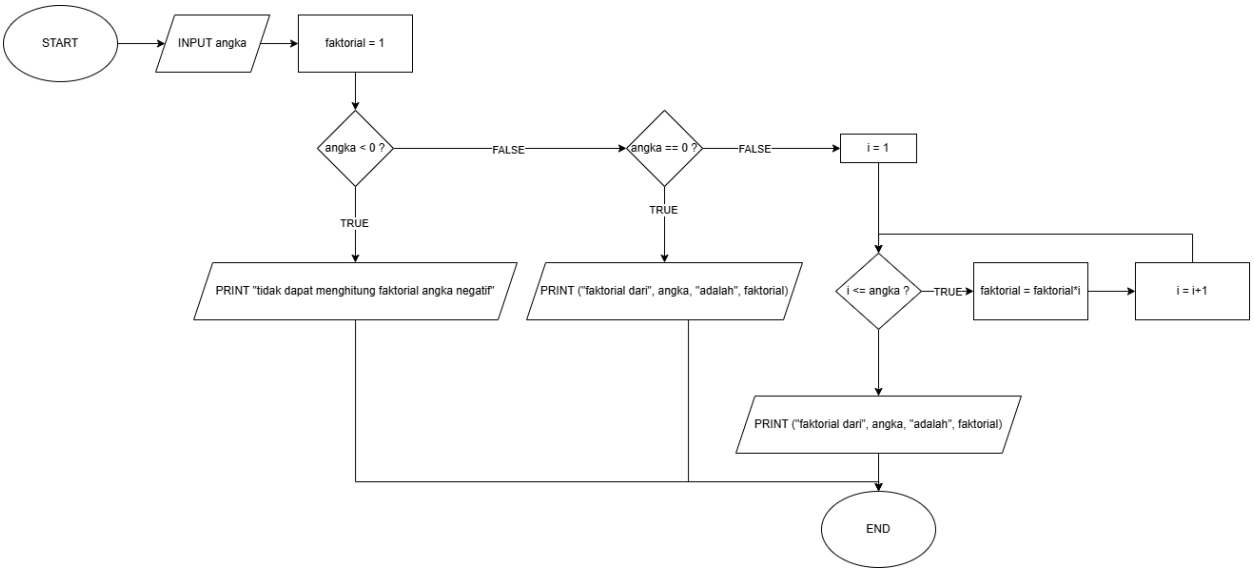
1. Algoritma dimulai (START).
2. User memberikan tiga nilai koefisien yaitu a, b, c (INPUT).
3. Pada percabangan pertama, program melakukan pengecekan apakah nilai koefisien a bernilai 0 (a == 0)? Jika TRUE, maka proses berhenti karena bukan merupakan persamaan kuadrat. Jika FALSE, maka proses berlanjut ke perhitungan.
4. Nilai diskriminan (D) dihitung dengan menggunakan rumus D = b**2 – 4*a*c.
5. Percabangan kedua, program melakukan pengecekan apakah nilai diskriminan lebih besar dari sama dengan 0 (D >= 0)? Jika TRUE, maka persamaan tersebut merupakan akar imajiner dan proses berakhir. Jika FALSE, maka persamaan tersebut merupakan akar real dan proses akan dilanjutkan.
6. Percabangan terakhir, program melakukan pengecekan apakah nilai diskriminan lebih besar dari 0 (D>0)? Jika TRUE, maka persamaan tersebut merupakan akar real dan berlainan lalu proses berakhir. Jika FALSE, maka persamaan tersebut memiliki akar real dan kembar lalu proses berakhir.
7. Algoritma selesai (END).

```
PROGRAM menentukanAkar
{program untuk menentukan jenis akar persamaan kuadrat berdasarkan koefisien a, b, dan c}

DEKLARASI
    int a, b, c, D

DESKRIPSI
    INPUT a, b, c
    IF a == 0 THEN
        PRINT "bukan persamaan kuadrat"
    ELSE
        D = (b**2) - (4 * a * c)
        IF D >= 0 THEN
            PRINT "akar real"
            IF D > 0 THEN
                PRINT "akar real dan berlainan"
            ELSE
                PRINT "akar real dan kembar"
            ENDIF
        ELSE
            PRINT "akar imajiner"
        ENDIF
    ENDIF
END
```

Alur ***pseudocode***:

1. Membuat sebuah header diawal algoritma.
2. Mendeklarasikan nama variabel dan tipe datanya (a, b, c, D memiliki tipe data integer).
3. Mulai alur:
    a. INPUT a, b, c: user mengisikan nilai koefisien a, b, c.
    b. Pengecekan koefisien a (IF a == 0 THEN): jika nilai a sama dengan 0, maka program akan menampilkan bukan persamaan linier.
    c. Perhitungan diskriminan (D) dan pemeriksaan akar: jika a tidak sama dengan 0, maka proses akan dilanjutkan dengan menghitung nilai diskriminan (D = b**2-4*a*c).
    d. Jika nilai diskriminan (D) lebih dari sama dengan 0 (D >= 0), maka akan ditampilkan bahwa (D) merupakan akar real. Selanjutnya, jika (D) lebih besar dari 0 (D > 0), maka akan ditampilkan bahwa (D) merupakan akar real dan berlainan.
    e. Jika (D) tidak lebih besar dari 0 (D = 0), maka akan ditampilkan bahwa (D) merupakan akar real dan kembar. Pengecekan terakhir, jika nilai (D) tidak lebih besar sama dengan 0 atau D < 0, maka akan ditampilkan bahwa nilai (D) merupakan akar imajiner.
    f. Program selesai ketika semua kondisi telah diperiksa (ENDIF, ENDIF, END).

SOAL 3



Alur ***flowchart***:

1. Algoritma dimulai (START).
2. User memberikan nilai angka (INPUT).
3. Inisialisasi variabel faktorial diatur menjadi 1.
4. Percabangan pertama, program mengecek apakah nilai angka yang dimasukan kurang dari 0 (angka < 0)? Jika TRUE, maka program akan menampilkan pesan “tidak dapat menghitung faktorial angka negatif” dan proses akan berakhir. Jika FALSE, maka program akan lanjut ke proses selanjutnya.
5. Percabangan kedua, program mengecek apakah nilai angka sama dengan 0 (angka == 0)? Jika TRUE, makan program akan menampilkan pesan faktorial dari 0 adalah 1 dan proses berakhir. Jika FALSE, maka program akan dilanjutkan.
6. Inisialisasi variabel i diatur menjadi 1.
7. Percabangan ketiga, program mengecek apakah i kurang dari sama dengan nilai angka (i <= angka)? Jika TRUE, maka nilai faktorial saat ini dikalikan dengan i (faktorial = faktorial*i) lalu nilai i ditambah 1 (i = i+1) dan proses mengulang (kembali) ke percabagan ketiga. Jika FALSE, maka program akan menampilkan hasil faktorial akhir dan proses berakhir.

8. Algoritma selesai (END).

```
PROGRAM hitungFaktorial
{program menghitung faktorial dari angka yang diberikan}

DEKLARASI
    int angka, faktorial, i

DESKRIPSI
    INPUT angka
    faktorial = 1

    IF angka < 0 THEN
        PRINT "tidak dapat menghitung faktorial angka negatif"
    ELSE IF angka == 0 THEN
        PRINT ("faktorial dari", angka, "adalah", faktorial)
    ELSE
        i = 1
        WHILE i <= angka DO
            faktorial = faktorial * i
            i = i + 1
        ENDWHILE
        PRINT ("faktorial dari", angka, "adalah", faktorial)
    ENDIF
END
```

Alur ***pseudocode***:

1. Membuat sebuah header diawal algoritma.
2. Mendeklarasikan nama variabel dan tipe datanya (angka, faktorial, i memiliki tipe data integer semua).
3. Mulai alur:
   a. INPUT angka: user akan memasukan suatu bilangan bulat yang akan disimpan di variabel angka.
   b. Menginisialisasi variabel faktorial menjadi 1.
   c. Pengecekan pertama (IF angka < 0 THEN): program akan memeriksa apakah nilai angka kurang dari 0, jika TRUE maka program akan menampilkan bahwa program tidak dapat menghitunh faktorial dari angka negatif. Jika FALSE, maka program akan memeriksa apakah angka sama dengan 0 (angka == 0) dan jika hasilnya TRUE, maka program akan menampilkan hasil faktorial dari 0 adalah 1.

d. Pengecekan kedua: jika nilai angka tidak memenuhi kedua kondisi diatas atau angka merupakan bilangan bulat positif, maka variabel I diinisialisasi menjadi 1.
e. Kemudian, program akan memulai kondisi perulangan WHILE yang akan terus berjalan selama nilai i kurang dari sama dengan angka (i <= angka). Jika nilai i kurang dari sama dengan angka, maka nilai faktorial sekarang dikalikan dengan nilai i (faktorial = faktorial*i) dan nilai i dijumlahkan dengan 1 (i = i+1).
f. Perulangan WHILE akan berakhir ketika nilai i lebih besar dari nilai angka (ENDWHILE).
g. Terakhir, program akan menampilkan hasil akhir yaitu nilai faktorial dari angka yang dimasukan (OUTPUT).
h. Program selesai ketika semua perhitungan sudah dilakukan (ENDIF, END).