

BAB IV

BLOK PERNYATAAN DAN PERCABANGAN SEDERHANA

4.1. Pengantar

Sejauh ini program yang Anda tulis akan dikerjakan oleh *compiler*urut dari atas ke bawah dan tidak ada satu perintahpun yang dilompati. Dalam prakteknya, tidak mungkin Anda menulis program yang strukturnya hanya urut saja dari atas ke bawah. Hal ini karena masalah yang harus diselesaikan oleh program sangatlah kompleks sehingga membutuhkan langkah program yang tidak mungkin sepenuhnya urut. Program Anda terpaksa harus melakukan pilihan-pilihan perintah untuk dikerjakan ketika program berjalan. Sebagai contoh, untuk menentukan berat ideal seseorang memakai rumus $berat\ ideal = tinggi - 100$, program Anda harus memberi tahu apakah seseorang memiliki berat ideal atau belum. Untuk bisa melakukan hal ini maka program harus mempunyai cara dalam memastikan apakah berat orang tersebut sama, lebih besar atau lebih kecil dari berat idealnya. Program harus mencetak pesan apakah orang tersebut terlalu kurus, sudah ideal beratnya atautkah malah terlalu gemuk. Dengan kata lain, program harus mempunyai kemampuan untuk memilih pesan manakah yang tepat untuk ditampilkan. Hal ini tidak mungkin dilakukan kalau pernyataan dalam program hanya dikerjakan secara runtut dari atas ke bawah. Compiler harus diberi tahu apakah pesan “terlalu kurus”, “sudah ideal” atau “terlalu gemuk” yang harus ditampilkan di layar. Cara memberi tahu *compiler* untuk melakukan hal inilah yang disebut dengan **perintah percabangan**. Dengan perintah ini *compiler* akan mengerjakan beberapa perintah jika memenuhi syarat tertentu dan akan melompatinya jika syarat tersebut tidak dipenuhi.

4.2. Blok Pernyataan

Sebelum Anda menulis program yang memuat perintah percabangan, ada hal mendasar yang harus Anda ketahui yakni **blok pernyataan**. Yang dimaksud dengan blok pernyataan adalah rangkaian atau gabungan beberapa pernyataan yang akan Anda perlakukan sebagai satu kesatuan. Blok pernyataan ini nantinya akan berfungsi seperti unit terkecil penyusun program Anda. Untuk

membedakan apakah suatu pernyataan/perintah menjadi bagian atau tidak dari suatu blok maka pernyataan-pernyataan tersebut diletakkan di antara dua kurung kurawal, yakni

```
{  
    pernyataan1;  
    pernyataan 2;  
    .....  
    pernyataan n;  
}
```

Sebagai contoh, berikut adalah dua buah blok program yang berbeda

```
{  
    System.out.print(" Nama Anda adalah : ");  
    System.out.println(nama);  
}  
  
{  
    // Blok ini menukar isi dua variabel x dan y  
    int tampunganSementara;    // tempat menampung sementara  
    tampunganSementara = x;    // simpan isi x ke tampungan sementara  
    x = y;                      // isi variabel x dengan isi dari y  
    y = tampunganSementara;    // kemudian isi y dengan tampungan sementara.  
}
```

Penulisan blok pernyataan dengan diawali dan diakhiri kurung kurawal dengan semua pernyataan ditulis dengan sedikit menjorok ke dalam tersebut menjadikan blok pernyataan tampak sebagai satu kesatuan dan mudah dibaca. Ini penting sekali karena dalam praktek nanti, program Anda akan tersusun dalam banyak blok pernyataan dan blok-blok tersebut bisa tersusun secara bertingkat yakni satu blok memuat beberapa blok dan masing-masing blok tersebut bisa saja memuat blok yang lain, demikian seterusnya. Oleh karena itu, Anda harus pandai-pandai menempatkan tanda kurung kurawal buka dan tutup di posisi yang tepat meskipun Netbeans juga

akan membantu Anda dalam penulisan blok pernyataan ini. Sebagai panduan praktis, jumlah kurung kurawal buka harus sama dengan jumlah kurung kurawal tutup. Dengan kata lain jumlah kurung kurawal haruslah genap. Dengan demikian struktur program Anda bisa jadi berbentuk seperti ini

```
{
  Pernyataan 1;
  Pernyataan 2;
  {
    Pernyataan 3;
    {
      Pernyataan 4;
      Pernyataan 5;
    }
    Pernyataan 6;
  }
  Pernyataan 7;
}
```

4.3. Percabangan Sederhana

Andaikan program Anda harus mengambil keputusan untuk menentukan apakah seorang mahasiswa lulus atau tidak dari suatu matakuliah. Aturan yang digunakan dalam hal ini adalah jika nilai lebih dari atau sama dengan 65 maka mahasiswa tersebut lulus dan jika nilainya kurang dari 65 maka ia tidak lulus. Andaikan Anda mempunyai variabel nilai yang menyimpan nilai matakuliah mahasiswa tersebut maka secara kasar struktur program Anda haruslah berbentuk

```
jika (nilai >= 65) {
    cetak tulisan “lulus”;
}
jika tidak {
    cetak tulisan “tidak lulus”;
}
```

Struktur program di atas menjadi berbentuk demikian di Java

```
if (nilai >= 65 ) {  
    System.out.println("Anda lulus.");  
}  
else {  
    System.out.println("Anda tidak lulus.");  
}
```

Perhatikan bahwa dalam program di atas ada 2 blok pernyataan yakni

```
{  
    System.out.println("Anda lulus.");  
}  
dan  
  
{  
    System.out.println("Anda tidak lulus.");  
}
```

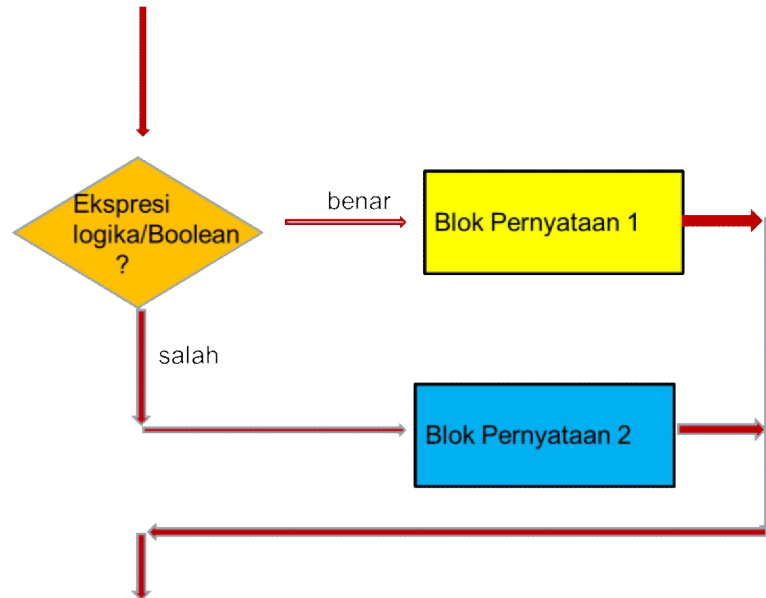
Kedua blok pernyataan ini tidak akan semuanya dieksekusi oleh *compiler* tetapi hanya salah satu saja dan hal itu tergantung apakah isi variabel nilai lebih dari 65 atau tidak. Jika memang isi variabel nilai lebih dari 65 maka blok pernyataan yang pertama yang akan dieksekusi tetapi jika misalnya isi variabel nilai hanya 50 maka blok pernyataan yang kedua yang akan dieksekusi. Dengan kata lain, apabila pernyataan nilai ≥ 65 bernilai benar maka blok pertama yang dikerjakan dan apabila nilai ≥ 65 bernilai salah maka blok kedua yang dikerjakan compiler.

Secara umum pengambilan keputusan dalam program dilakukan dengan memakai perintah

```
if (ekspresi logika/boolean)  
    { blok pernyataan 1 }  
else  
    { blok pernyataan 1 }
```

Blok pernyataan 1 akan dikerjakan apabila ekspresi logika yang mengikuti perintah **if** bernilai benar dan blok pernyataan 2 hanya dikerjakan apabila ekspresi logika tersebut bernilai salah.

Percabangan di atas juga dapat digambarkan memakai diagram alir (*flowchart*) berikut:



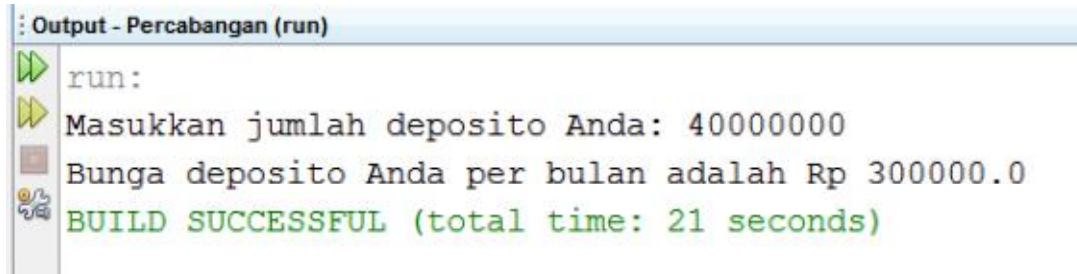
Ada kemungkinan Anda dihadapkan pada persoalan untuk mengeksekusi blok pernyataan pertama jika ekspresi logika benar dan bila salah lanjutkan saja ke perintah-perintah berikutnya. Artinya keadaan memang tidak menuntut adanya blok pernyataan kedua. Jika seperti ini situasinya maka Java memungkinkan Anda memberi perintah dengan struktur berikut

```
if (ekspresi logika/boolean)  
{ blok pernyataan 1 }
```

Berikut adalah program untuk menghitung bunga deposito yang besar bunganya berdasarkan besarnya tabungan. Jika tabungannya lebih dari 50 juta maka besarnya bunga 1 % per bulan sedang apabila tabungannya kurang dari 50 juta maka besar bunganya hanya 0,75 %.

```
1
2 package percabangan;
3
4 /**
5  *Program menghitung bunga bank
6  */
7 import java.util.Scanner;
8
9 public class BungaBank {
10     public static void main(String[] args) {
11
12         double deposito, bunga; // variable untuk menyimpan data deposito dan bunganya
13
14         Scanner dataBank = new Scanner(System.in);
15
16         System.out.print("Masukkan jumlah deposito Anda: "); //pemasukan data deposito
17         deposito = dataBank.nextDouble();
18
19         if (deposito >= 50000000){ // perhitungan bunga dng syarat besarnya deposito
20             bunga = 0.01 * deposito; // jika deposito lebih dari 50 juta
21         }
22         else {
23             bunga = 0.0075 * deposito; // jika deposito kurang dari 50 juta
24         }
25         System.out.println("Bunga deposito Anda per bulan adalah Rp "+ bunga);
26     }
27 }
```

Ketika dijalankan maka hasilnya adalah sbb



```
Output - Percabangan (run)
run:
Masukkan jumlah deposito Anda: 40000000
Bunga deposito Anda per bulan adalah Rp 300000.0
BUILD SUCCESSFUL (total time: 21 seconds)
```

Latihan

1. Buatlah program untuk mengecek apakah bilangan yang dimasukkan merupakan bilangan ganjil atau genap

4.4. Ekspresi Boolean

Untuk mengambil keputusan atau menarik kesimpulan dari pernyataan if di atas, maka diperlukan pengecekan terhadap ekspresi boolean / logika. *Apakah yang dimaksud dengan ekspresi boolean ? Ekspresi boolean adalah ekspresi yang mengandung nilai kebenaran.* Contoh ekspresi Boolean yang dimengerti oleh komputer adalah :

1. `((suka == 'Matematika') && (hobi == 'Komputer'))`
2. `((no_rumah == 46) && (warna == 'putih'))`
3. `(bil % 2 == 0)`

Jika bil bernilai 4 maka ekspresi Boolean tersebut mempunyai nilai kebenaran *true* atau 1. Hasil dari setiap ekspresi boolean akan bernilai benar atau salah. Dalam bahasa pemrograman Java, nilai *benar* akan dinyatakan dengan *true*, dan nilai *salah* akan dinyatakan dengan *false*. Untuk membentuk ekspresi Boolean, diperlukan operator relasional dan operator logikal. Kita mengulang dari bab sebelumnya tentang operator relasional dan operator logikal yang dipakai dalam bahasa pemrograman Java

4.4.1. Operator Relasional

Operator relasional adalah operator yang membandingkan beberapa ekspresi.

Operator	Arti
==	Sama dengan
>	Lebih besar dari
>=	Lebih besar atau sama dengan
<	Kurang dari
<=	Kurang atau sama dengan
!=	Tidak sama dengan

4.4.2. Operator Logikal

Operator logikal adalah operator yang menggabungkan ekspresi yang mengandung operator relasional.

Operator	Arti	Keterangan
&&	AND / DAN	Bernilai 1 hanya jika ekspresi pada kedua sisi bernilai 1
	OR / ATAU	Bernilai 1 jika satu atau kedua ekspresi bernilai 1
!	NOT / TIDAK	Jika ekspresi bernilai 1 maka hasil akhir bernilai 0, sebaliknya jika ekspresi bernilai 0 maka hasil akhirnya bernilai 1.

Tabel Kebenaran operator logikal untuk satu atau dua ekspresi

Ekspresi 1(p)	Ekspresi 2 (q)	(p && q)	(p q)	! p
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Contoh :

Ekspresi boolean ((a > 5) && (a <= 10))

Nilai a	(a > 5)	(a <= 10)	(a > 5) && (a <= 10)
7	1	1	1
12	1	0	0
10	1	1	1
3	0	1	0

4.4.3. Ekuivalensi Tipe

Dalam sebuah ekspresi Boolean, pernyataan-pernyataan yang dibandingkan tersebut harus mempunyai tipe data yang sama.

Contoh :

1. Jika variabel A = '5' maka :

Ekspresi Boolean (A > 5) merupakan ekspresi Boolean yang tidak sah, karena variabel A bertipe char (character), sedangkan 5 bertipe int (integer).

2. Jika variabel A = '5' maka :

Ekspresi Boolean (A == 'R') merupakan ekspresi Boolean yang sah, karena variabel A bertipe char, sedangkan 'R' juga bertipe char.

4.4.4. Order/Urutan Operator

Jika terdapat lebih dari satu operator dalam satu ekspresi, maka urutan/*order* pengerjaan dari setiap operator adalah sebagai berikut:

Urutan	Operator
1	()
2	*, /, %
3	+, -
4	!
5	>, >=, <, <=
6	==, !=
7	&&
8	

Contoh :

```
public static void main()
{
    int a= 3, b = 23;
    boolean y;
    y = !a > 9;
    z= !(a > 9) ;
    System.out.println("Hasil dari ekspresi pertama adalah "+y);
    System.out.println("Hasil dari ekspresi kedua adalah "+z);

}
```

1. Hasil $y = !a > 9$ adalah 0 karena :

Ekspresi	Evaluasi	Nilai kebenaran
$!a$	$!3$	0
$!a > 9$	$0 > 9$	0

2. Hasil $z = !(a > 9)$ adalah 1 karena :

Ekspresi	Evaluasi	Nilai kebenaran
$A > 9$	$3 > 9$	0
$!(a > 9)$	$!(0)$	1

Hati-hati !

Operator relasi `==` sering keliru dengan operator penugasan `=`.

Contoh dibawah ini adalah kedua ekspresi yang benar tetapi mengandung arti yang berbeda :

`int y = 3;`

`x = ((y > 2) && (y == 4))` \Rightarrow nilai x adalah false

`x = ((y > 2) && (y = 4))` \Rightarrow nilai x adalah true

Mengapa ?

4.5. Latihan

1. Carilah nilai kebenaran ekspresi Boolean berdasarkan statemen berikut :

`int x = 3;`

`int y = 5;`

`int z = 2;`

`int u = 3;`

- `(x == u)` bernilai true/1
- `x != u`
- `!(x >= u) && (y != z)`
- `(y == x) || (u > z)`
- `x + 2 < 10 && y - 2 < 5 || u + 4 < z`

4.6. Gabungan if

Setelah mempelajari statemen *if* dan *if - else*, maka anda harus bijaksana dalam menggunakannya. Gunakanlah statemen *if* untuk mengambil keputusan yang sederhana dan gunakanlah statemen *if - else* untuk memilih satu dari dua buah kemungkinan.

Contoh :

```
import java.util.Scanner;

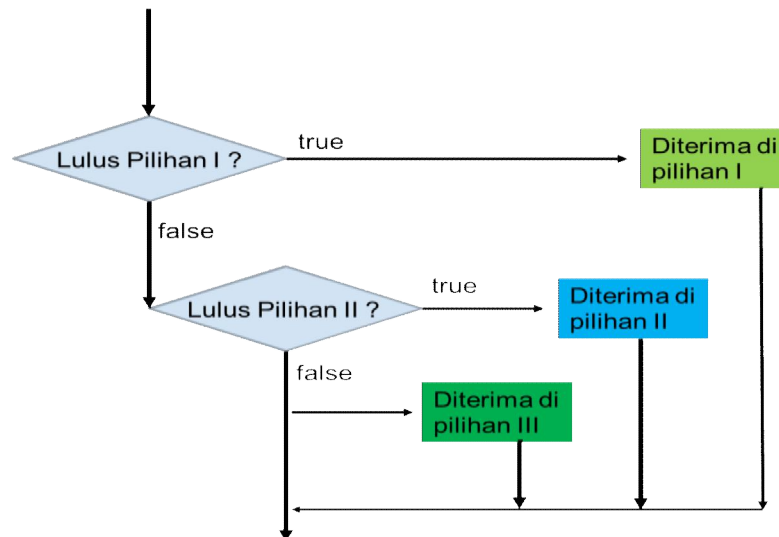
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    int jumPelanggan;
    double totalJual;
    System.out.println("Berapa banyak pelanggan ?");
    jumPelanggan=sc.nextInt();
    System.out.println("Berapa total penjualan ?");
    totalJual=sc.nextDouble();
    if (jumPelanggan > 25) // if pertama, memuat hanya 1 perintah, tak perlu pakai { }
        System.out.println("Pesan makanan lebih banyak untuk besok \n");

    if (totalJual >= 10000000) { //if-else kedua, tidak berhubungan dengan if pertama
        System.out.println("Cek kembali stock \n");
        System.out.println("Beri penghargaan untuk sales \n");
    }
    else {
        System.out.println("Pecat sales \n");
    }

    if ((jumPelanggan >= 50) && (totalJual >= 100000000)) { //if ketiga, berdiri sendiri
        System.out.println("Liburkan penjualan \n");
        System.out.println("Cek kembali stock \n");
    }
}
```

4.7. Percabangan Bertingkat

Seringkali pengambilan keputusan yang harus dilakukan oleh program berlangsung secara bertingkat. Sebagai contoh dalam penerimaan mahasiswa baru, calon mahasiswa mempunyai 3 pilihan program studi yakni pilihan 1, pilihan 2 dan pilihan 3. Seorang calon mahasiswa bisa diterima di pilihan 1 (tentu saja !) tetapi ia hanya bisa diterima di pilihan 2 kalau tidak diterima di pilihan 1 dan baru bisa diterima di pilihan 3 kalau tidak diterima di pilihan 1 dan pilihan 2. Situasi pengambilan keputusan ini dapat digambarkan memakai diagram alir seperti di bawah ini.



Di java alur di atas menjadi seperti berikut:

```
if ( lulusPilihan1) {  
    diterima = pilihan1;  
}  
else if (lulusPilihan2) {  
    diterima = pilihan2;  
}  
else {  
    diterima = pilihan3;  
}
```

Contoh:

Persamaan kuadrat yang berbentuk $ax^2 + bx + c = 0$ mempunyai tiga bentuk penyelesaian tergantung kepada besarnya nilai $D = b^2 - 4ac$. Jika $D < 0$ maka persamaan kuadrat tersebut tidak mempunyai akar yang real, sedang jika $D = 0$ maka hanya mempunyai satu penyelesaian/akar dan jika $D > 0$ maka akarnya ada 2 buah. Secara umum apabila $D \geq 0$ maka akarnya dapat dicari memakai rumus berikut

$$x_{1,2} = (-b \pm \sqrt{D}) / (2a)$$

dengan kata lain $x_1 = (-b + \sqrt{D}) / (2a)$ dan $x_2 = (-b - \sqrt{D}) / (2a)$. Berikut program untuk mencari dan menentukan jenis akar persamaan kuadrat tersebut.

```
import java.util. Scanner;
public class PersamaanKuadrat {
    public static void main(String[] args) {
        double a, b, c, D, x1, x2;
        Scanner dataKuadrat = new Scanner(System.in);

        // Baca data koefisien persamaan dari keyboard
        System.out.println(" Koefisien x2 (a) : ");
        a = dataKuadrat.nextDouble();
        System.out.println(" Koefisien x (b) : ");
        b = dataKuadrat.nextDouble();
        System.out.println(" Konstanta (c) : ");
        c = dataKuadrat.nextDouble();

        // Hitung nilai D
        D = b*b - (4*a*c);
        // Hitung akar sesuai nilai D
        if (D < 0){ // tidak punya akar real
            System.out.println("Tidak mempunyai akar nyata/real");
        }
    }
}
```

```

else if (D == 0 ) { // akarnya sama
    x1 = -b/(2*a);
    System.out.println("Akarnya tunggal yakni : "+x1);
}
else { // akarnya dua berbeda
    x1 = (-b + Math.sqrt(D))/(2*a);
    x2 = (-b - Math.sqrt(D))/(2*a);
    System.out.println("Akar pertama : "+x1);
    System.out.println("Akar kedua : "+x2);
}
}
}
}

```

4.8. Praktikum

1. Buat program yang dapat mengurutkan 3 bilangan real (tipe double/float) sembarang yang dimasukkan lewat keyboard. Misal yang dimasukkan user adalah sbb:

Bilangan 1: 5.4

Bilangan 2: 3.5

Bilangan 3: 7.9

Urutan 3 bilangan di atas dari kecil ke besar adalah 3.5, 5.4, 7.9

2. Buat program yang dapat menentukan jenis segitiga yakni apakah suatu segitiga merupakan segitiga sama sisi, sama kaki, siku-siku atau segitiga sembarang berdasarkan panjang ketiga sisinya yang dihitung melalui jarak antara dua titik segitiga. Titik sudut segitiga yang berupa absis dan ordinat yakni (x1,y1), (x2,y2), dan (x3,y3) masing-masing nilainya dimasukkan lewat keyboard. Segitiga adalah sama sisi jika semua sisinya mempunyai panjang yang sama. Segitiga sama kaki apabila ada 2 sisinya yang sama panjang dan segitiga siku-siku bila memenuhi teorema Pitagoras yakni jika a, b, dan c sisi segitiga maka $a^2 = b^2 + c^2$ di mana a adalah sisi terpanjang sementara b dan c adalah 2

sisi yang lain. Segitiga yang tidak memenuhi salah satu di atas disebut dengan segitiga sembarang. Jadi input dari program Anda adalah besarnya absis dan ordinat ketiga segitiga dan outputnya adalah jenis segitiga.

3. Buat program untuk menentukan bintang (zodiac) seseorang berdasarkan tanggal lahirnya yang dimasukkan lewat keyboard. Format tanggal yang Anda gunakan cukup berupa tanggal dan bulan yang berupa bilangan bulat. Program Anda selain dapat menentukan nama bintang seseorang juga dapat menampilkan sifat-sifat pokok dari bintang tersebut yang adapat Anda cari lewat Google. Contoh input dan output progra Anda adalah sbb:

Nama : Edi
Alamat : Jln. Colombo Purwokerto
Tanggal lahir : 20
Bulan lahir : 5
Anda berbintang : Taurus
Sifat Anda : Murah hati dan penyanyang.....dst dst

- Aquarius : 20 Januari-18 Februari
- Pisces : 19 Februari-20 Maret
- Aries : 21 Maret-19 April
- Taurus : 20 April-20 Mei
- Gemini : 21 Mei-20 Juni
- Cancer : 21 Juni-22 Juli
- Leo : 23 Juli-22 Agustus
- Virgo : 23 Agustus-22 September
- Libra : 23 September-22 Oktober
- Scorpio : 23 Oktober-21 November
- Sagitarius: 22 November-21 Desember

4. Buat program yang dapat menentukan apakah seseorang memiliki berat badan ideal, terlalu kurus atau terlalu gemuk berdasarkan tinggi badan (cm) dan berat badan (kg) yang dimasukkan lewat keyboard. Aturannya adalah :

Jika $90 \leq (\text{tinggi badan} - \text{berat badan}) \leq 110$ maka berat badan ideal

Jika $(\text{tinggi badan} - \text{berat badan}) < 90$ maka terlalu gemuk

Jika $(\text{tinggi badan} - \text{berat badan}) > 110$ maka terlalu kurus

5. Modifikasi program menghitung akan persamaan kuadrat sehingga dapat menampilkan semua jenis akar yang mungkin termasuk menampilkan pesan bila tidak mempunyai akar yang rasional
6. Buat program untuk menentukan nilai final (berupa huruf) mahasiswa berdasarkan nilai uts1, uts2 dan uas yang dimasukkan lewat keyboard. Nilai total dihitung memakai rumus $\text{nilai total} = 30 \% \times \text{uts1} + 30 \% \times \text{uts2} + 40 \% \times \text{uas}$. Sedang nilai final ditentukan berdasarkan kriteria berikut:

A : nilai total ≥ 80

B : $65 \leq \text{nilai total} < 80$

C : $55 \leq \text{nilai total} < 65$

D : $50 \leq \text{nilai total} < 55$

E : nilai total < 50

7. Bonus akhir tahun karyawan ditentukan berdasarkan prosentase dari gaji pokok dengan memakai rumus berikut

Golongan	Masa Kerja (tahun)		
	0 – 10	11 - 20	21 - 30
1	50 %	60 %	70 %
2	60 %	70 %	80 %
3	70 %	80 %	90 %
4	80 %	90 %	100 %

Buat program yang dapat menghitung bonus karyawan berdasarkan gaji pokok, golongan serta masa kerja yang dimasukkan lewat keyboard.