No. 0818-0824-2511

Date:

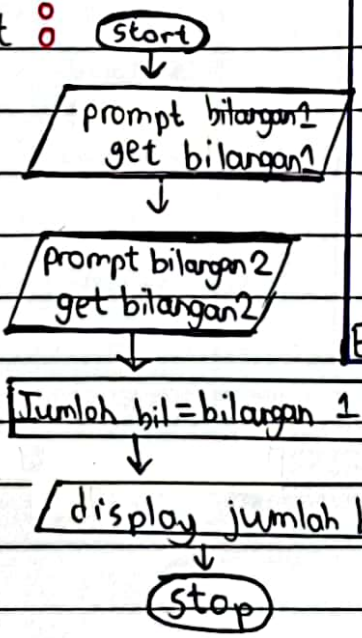
IPO = Input Process Output

- start /stop (Terminal symbol)
- input /output symbol
- predefined process symbol
- Decision symbol
- Flowlines
- Connector symbol
- Off page connector symbol
- Process symbol

Identifikasi problem :
Input : Bilangan 1
Bilangan 2
Proses : Jumlah bil = Bilangan 1 + Bilangan 2
Output : Jumlah bil

↳ Pseudo code : program utama
prompt bilangan 1
get bilangan 1
prompt bilangan 2
get bilangan 2
jumlah bil = bilangan 1 + bilangan 2
display jumlah bil
END

↳ Flowchart :



No.

Date:

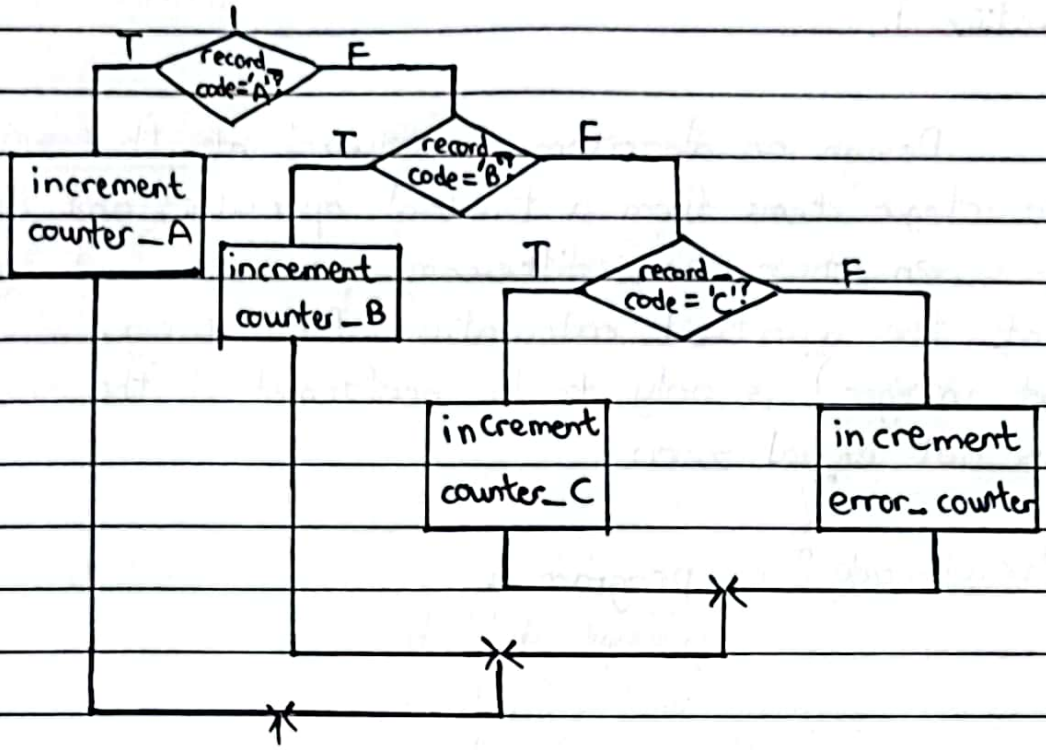
↳ Program C :

```c
# include <stdio.h>
int main () {
    int bilangan 1, bilangan 2, jumlahBil;
    scanf ("%d", & bilangan 1);
    scanf ("%d", & bilangan 2);
    jumlahBil = bilangan 1 + bilangan 2;
    printf ("%d", & jumlahBil);
    return 0;
}
```

↳ Program C++ :

```cpp
# include <iostream>
using namespace std;
int main () {
    int bilangan 1, bilangan 2, jumlahBil;
    cin >> bilangan 1;
    cin >> bilangan 2;
    jumlahBil = bilangan 1 + bilangan 2;
    cout << jumlahBil;
    return 0;
}
```

T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F
A	B	And	Or



↳ Pseudo code

```
IF (record_code = 'A') THEN
    increment counter_A
ELSE
    IF (record_code = 'B') THEN
        increment counter_B
    ELSE
        IF (record_code = 'C') THEN
            increment counter_C
        ELSE
            increment error_counter
        ENDIF
    ENDIF
ENDIF
```

Practice 1

Input	Processing	Output
bil1 bil2	prompt for bil1 prompt for bil2 accept bil1 accept bil2 compute sum of bil1&bil2 compute difference of bil1 &bil2 compute quotient of bil1 & bil2 (If bil2 is zero, then print quot undefined) compute product of bil1 &bil2	hasil j hasil k hasil b hasil x

Practice 2

Input	Processing	Output
nim score	prompt for score accept score match the score	grade

Practice 3

Input	Processing	Output
term_op pri_cod	prompt for pri_cod accept pri_cod match pri_cod compute discount and discounted price	term_op disc_price new_disc_price Invalid pricing code No discount

No.

Date

Practice 1

Design an algorithm in flowchart that will receive two integer items from a terminal operator, and display to the screen their sum, difference, product, and quotient. Note that the quotient calculation (first integer divided by second integer) is only to be performed if the second integer does not equal zero.

↳ Pseudocode :

```
program utama                                   24 byte
    prompt bil1, bil2
    get    bil1, bil2
    set    hasil j=0, hasil k=0, hasil b=0, hasil x=0
    calculate hasil j = bil 1+bil2
    calculate hasil k = bil 1 - bil2
        IF (bil 2 != 0) THEN
            calculate hasil b = bil 1/bil2
        ELSE
            set hasil b = 0
        ENDIF
    calculate hasil x = bil 1 * bil2
    display hasil j
    display hasil b
    display hasil k
    display hasil x
END
```

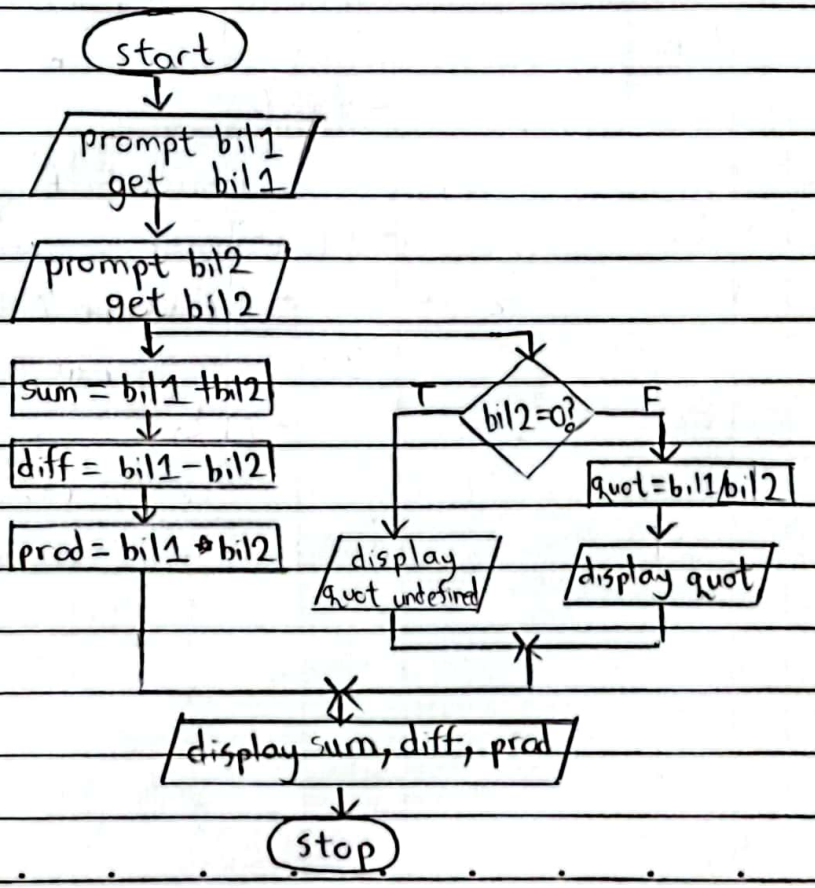
No.

Date:

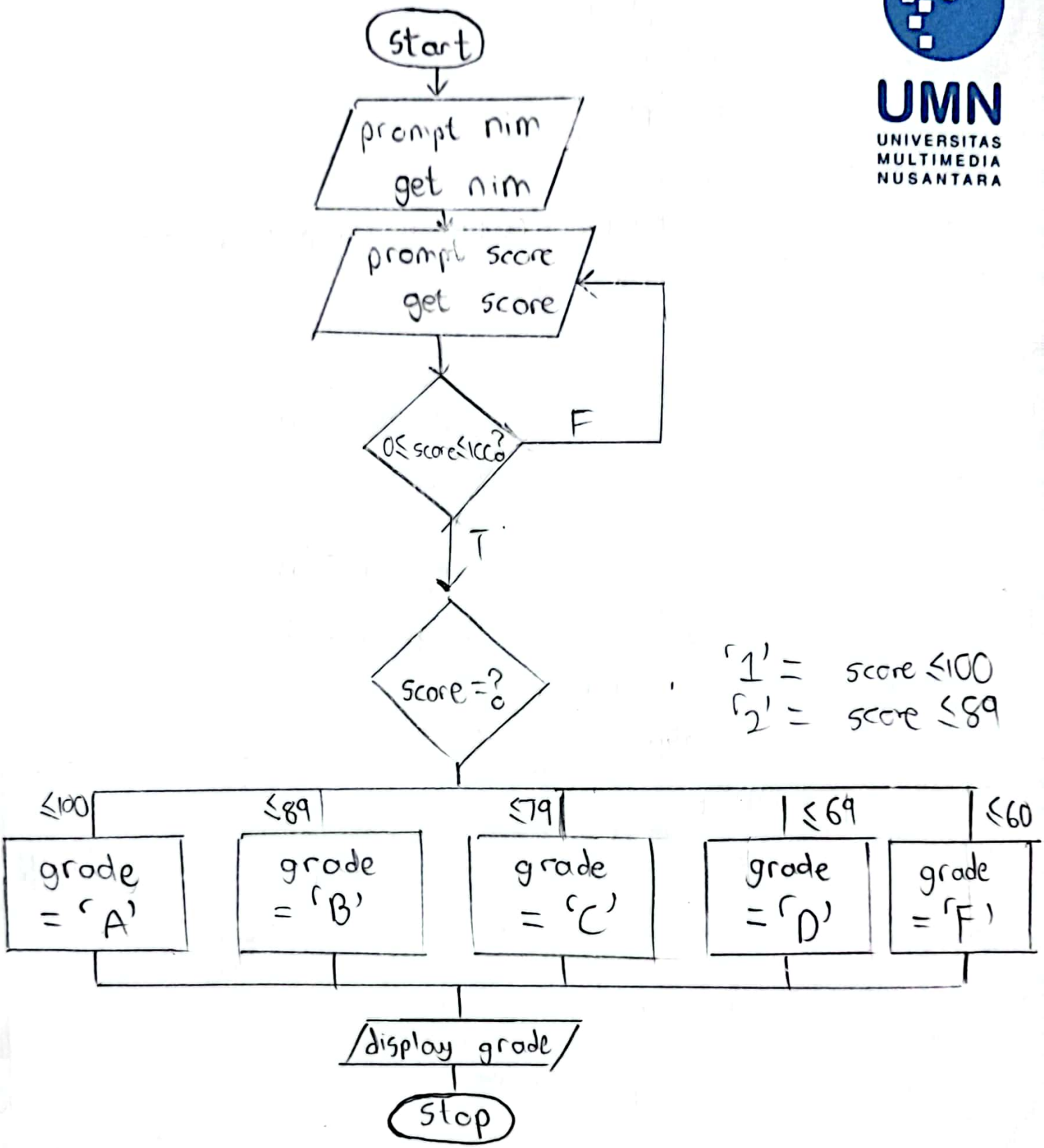
↳ Pseudocode :

```
program utama                    12 byte
    prompt bil1, bil2
    get bil1, bil2
    set hasil = 0
    hasil = bil1 + bil2
    display hasil
    hasil = bil1 - bil2
    display hasil
    hasil = bil1 * bil2
    display hasil
    IF (bil2 != 0) THEN
        hasil = bil1 / bil2
    ELSE
        hasil = 0
    ENDIF
    display hasil
END
```

↳ Flowchart :



Practice 2



No.

Date:

Practice 2

Design an algorithm in flowchart that will prompt an operator for a student's serial number and the student's exam score out of 100. Your program is then to match the exam score to a letter grade and print the grade to the screen. Calculate the letter grade as follows :

Exam Score	Assigned Grade
90 and above	A
80-89	B
70-79	C
60-69	D
below 60	E

↳ Pseudocode :

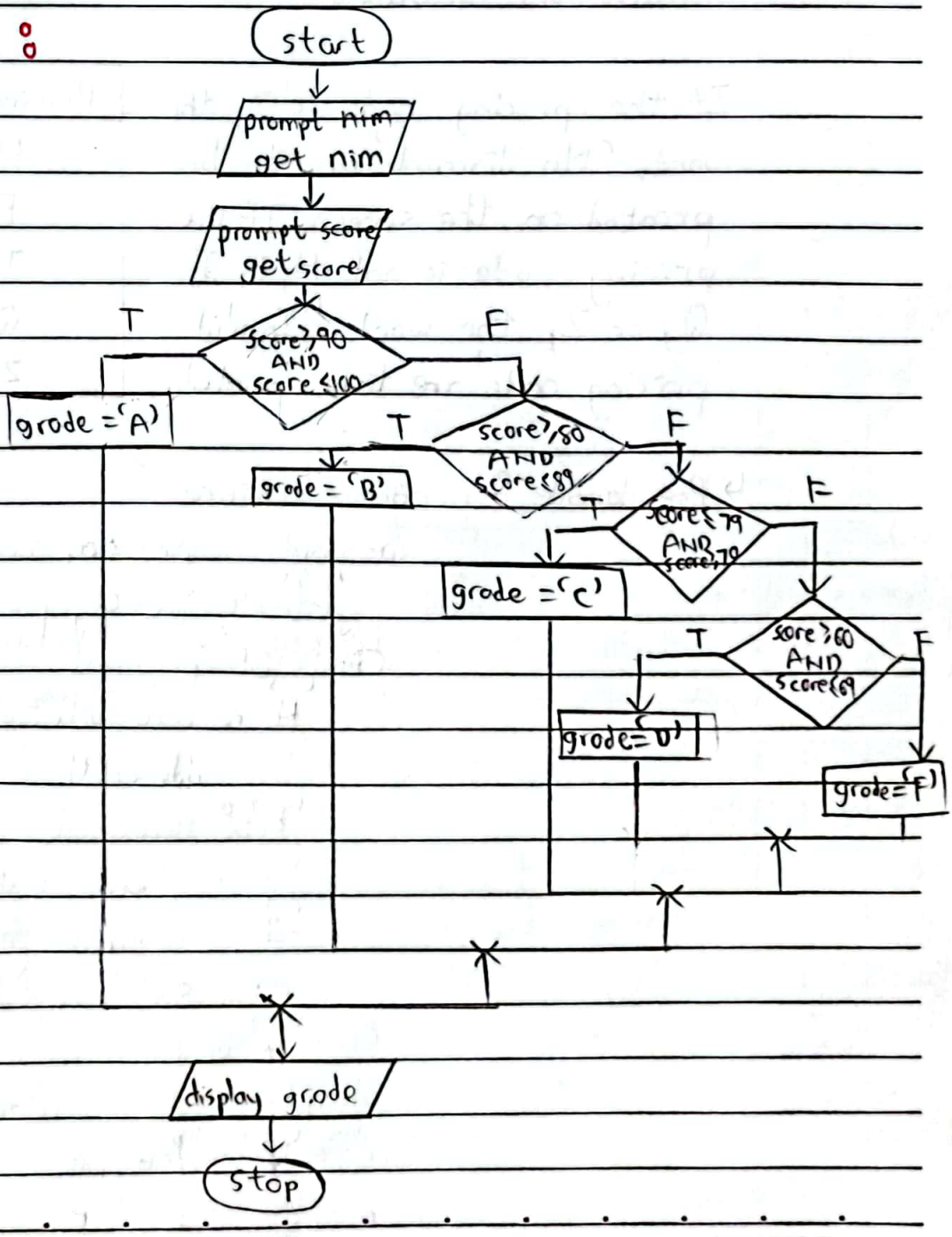
```
program utama
    prompt nim, score
    get nim, score
    IF (score >= 90 AND score <= 100) THEN
        grade = 'A'
    ELSE
        IF (score >= 80 AND score <= 89) THEN
            grade = 'B'
        ELSE
            IF (score >= 70 AND score <= 79) THEN
                grade = 'C'
            ELSE
                IF (score >= 60 AND score <= 69) THEN
                    grade = 'D'
                ELSE
                    IF (score >= 0 AND score < 60) THEN
```

No.

Date:

```
        grade = 'F'
    ENDIF
  ENDIF
ENDIF
ENDIF
ENDIF
display grade
END
```

↳ Flow chart :



No.

Date:

Practice 3

Design an algorithm in flowchart that will prompt a terminal operator for the price of an article and a pricing code. Your program is then to calculate a discount rate according to the pricing code and print to the screen the original price of the article, the discount amount, and the new discounted price. Calculate the pricing code and accompanying discount amount as follows:

If the pricing code is Z, the words 'No discount' are to be printed on the screen. If the pricing code is not H, F, T, Q, or Z, the words 'Invalid pricing code' are to be printed.

Pricing Code	Discount Rate
H	50%
F	40%
T	33%
Q	25%
Z	0%

↳ Pseudocode :

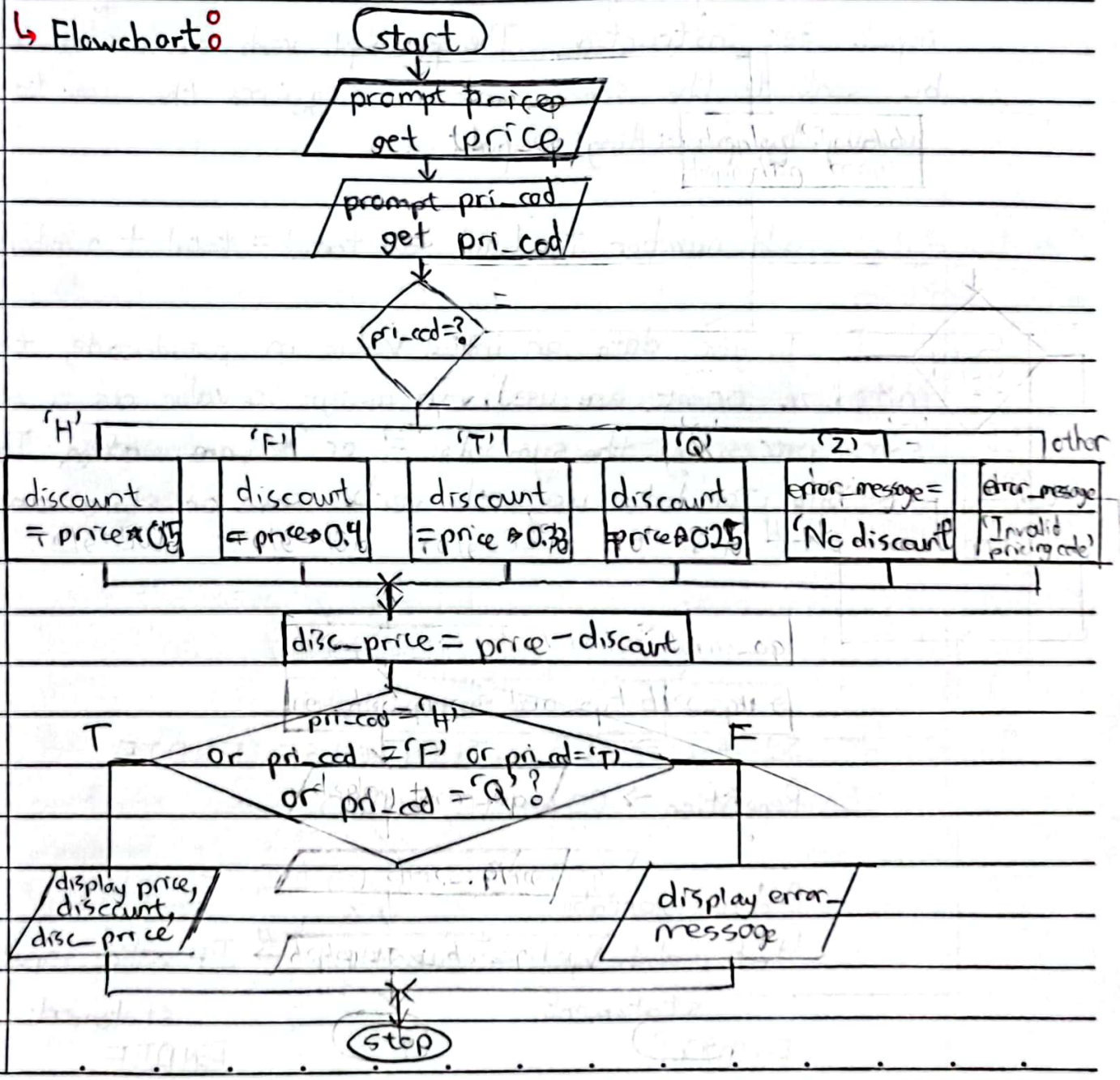
```
program utama
    prompt 'price, pri_cod'
    get price, pri_cod
    CASE OF pri_cod
        'H' : discount = price * 0.5
              break
        'F' : discount = price * 0.4
              break
        'T' : discount = price * 0.33
              break
        'Q' : discount = price * 0.25
              break
        'Z' : discount = 0
              error_message = 'No Discount'
```

No.

Date:

```
        break
    other: error_message = 'Invalid Pricing Code'
ENDCASE
disc_price = price - discount
IF(pri_cod = 'H' or pri_cod = 'F or pricod = 'T' or pri_cod = 'Q')THEN
    display price, discount, disc_price
ELSE
    display error_message
ENDIF
END
```

↳ Flowchart:



No.

Date:

Read is used when the algorithm is to receive input from a record on a file. Get is used when the algorithm is to receive input from the keyboard.

Print is usually used when the output is to be sent to the printer. Write is used when the output is to be written to a file. If output is to be written to the screen, the put, output, or display are used in algorithm.

Usually output prompt instruction is required before an input Get instruction. The prompt verb causes a message to be sent to the screen, which requires the user to respond, usually by providing input.

add number to total ⇔ total = total + number

To give data an initial value in pseudocode, the verbs initialize or set are used. To assign a value as a result of some processing, the symbols '=' or '←' are written. To keep a variable for later use, the verbs save or store are used.

↳ hasil = 0
↳ initialize hasil to 0
↳ set hasil to 0

nested selection → nested IF statement
combined selection → IF & AND, OR

Selection → IF ... THEN, ELSE, ENDIF
Repetition → DO WHILE, ENDDO

Boolean varrables
↳ IF valid_input = true THEN → IF valid_input THEN
statement statement
ENDIF ENDIF

Assignment =
Arithmetic + - * / % ( )
Logical And Or Not Bitwise
Relation = < > >= <= <> → Not equal to

No.

Date:

2 kinds of repetition :

- Sentinel - controlled repetition (Gtw kpn stop) = Indefinite repetition
- Counter - controlled repetition (Tw kpn stop) = Definite repetition = For statement

A loop is a group of instructions the computer executes repeatedly while some loop repetition condition remains true.

Sentinel values are used to control repetition when :

- The precise number of repetitions is not known in advance
- The loop includes statements that obtain data each time the loop is performed.

Sentinel - controlled : while, Do-while, Do-until.

Both While and Do-while loops cause a statement or set of statements to repeat as long as a condition is true.

Gada di C

Do-until loop causes a statement or set of statements to repeat until a condition is true.

- While a condition is true, do some task
  - ↳ The loop is repeated when the condition is true (When its value isn't 0)

The Do-while loop is a post test loop. This means it performs an iteration before testing its condition. As a result, the Do-while loop always performs at least one iteration, even if its condition is false to begin with.

operator precedence

() ↓↑ peringkat operator

Big BOSS

No.

Date:

Sometimes, it's more convenient to write a loop that iterates until a condition is true - that is, a loop that iterates as long as condition is false

A loop variable of counter-controlled repetition is used to count the number of repetitions.

1. Initialization : Loop control variable is set to an initial value before the while statement is reach
2. Testing : Loop control variable is tested before the start of each loop repetition.
3. Updating/Increment : Loop control variable is updated (Incremented/decremented) during each iteration.

Practice 1

Design an algorithm in pseudocode which displays the numbers 1 through the maximum value (User input) and their squares. Maximum value = 5.

Number	Square
1	1
2	4
3	9
4	16
5	25

No.

Date:

↳ program 1

```
declare integer x, hasil
set x = 1
prompt max
  get max
display "Number    Square"
WHILE ( x <= max)
    display x, "    ", x * x
    max = max + 1
ENDWHILE
END
```

Practice 2

Design an algorithm in pseudocode that print the following sequence of values 20, 14, 8, 2, -4, -10

↳ program 2

```
declare integer x, number
set x = 1
set number = 20
FOR x = 1 to 6
    display number
    x = x + 1
    number = number - 6
ENDFOR
END
```

No.

Date

Practice 3

Design an algorithm in pseudocode that print the following sequene of values 19, 27, 34, 40, 45

```
program 3
    declare integer x, hasil
    set x = 1
    set hasil = 19
    FOR x = 1 to 5
        display hasil
        hasil = hasil + 9 - x
        x = x + 1
    ENDFOR
END
```

Practice 4

The factorial function is used frequently in probability problems. The factorial of a positive integer n (Written n! and pronounced "n factorial") is equal to the product of the positive integers from 1 to n. Write in a pseudocode that evaluates the factorials of the integers from p to q (p and q are inputted by user). The screen dialogue should appear as follows :

1 5
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120

3 8
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320

No.

Date:

4 program 4

```
declare integer : a, b, awal, akhir
prompt awal, akhir
get awal, akhir
  a = awal
FOR  a = awal to akhir
   display  a, "! =" 
   fakto = 1
     b = 1
   FOR b  = 1 to a
      fakto = fakto * b
      b = b + 1
   ENDFOR
   display fakto
   a = a + 1
ENDFOR
END
```

No.

Date:

1. Jika input = 6, buat tampilan angka berikut 1, 22, 333, 4444, 5555, 666666.

c) program_utama
```
declare integer ulang, n, cetak
set ulang = 1
prompt n
get n
WHILE (ulang <= n)
    cetak = 1
    WHILE (cetak <= ulang)
        display ulang
        cetak = cetak + 1
    ENDWHILE
    display " "
    ulang = ulang + 1
ENDWHILE
END
```

2. Jika input = 6, buat tampilan angka berikut 666 666, 5555, 4444, 333, 22, 1

c) progran_kedua
```
declare integer ulang, n, cetak
set ulang = n
prompt n
get n
WHILE (ulang >= 1)
    cetak = 1
    WHILE (cetak <= ulang)
        display ulang
```

No.

Date:

```
            setak = setak+1
        ENDWHILE
        display " "
        ulang = ulang-1
    ENDWHILE
END
```

3. Jika input = 10, buat tampilan angka berikut 1, 2, 4, 8, 16, 32, 64, 128, 256, 512

→ c) program_ketiga

```
    declare integer n, bil, hsl
    prompt n
    get n
    set bil=1, hsl=1
    FOR bil = 1 to n
        display hsl, " "
        hsl = hsl * 2
        bil = bil+1
    ENDFOR
END
```

4. Jika input = 1 dan 10, tampilkan bilangan yang habis dibagi 3 dari n_awal=1 hingga n_akhir = 10, contoh: 3 6 9

→ c) program_keempat

```
    declare integer awal, akhir, hsl
    prompt awal, akhir
    get awal, akhir
    FOR hsl = awal to akhir
        IF (hsl % 3 = 0) THEN
```

No.

Date:

```
            display hsl, " "
        ENDIF
        hsl = hsl + 1
    ENDFOR
END
```

5. Masukkan 10 bilangan dengan looping, kemudian cari bilangan terbesar dan bilangan terkecil dari 10 bilangan tersebut

c)

```
program_kelima
    declare integer max, min, i
    max = MIN_INT
    min = MAX_INT
    set i = 1
    FOR i = 1 to 10
        prompt bil
        get bil
        IF (bil > max) THEN
            max = bil
        ENDIF
        IF (bil < min) THEN
            min = bil
        ENDIF
        i = i + 1
    ENDFOR
    display " terbesar = ", max
    display " terkecil = ", min
END
```

No.

Date:

6. Masukkan 10 bilangan dengan looping, kemudian hitung jumlah bilangan genap dari 10 bilangan tersebut.

→ c) program_kecram

```
declare integer n, i, genap
set i = 1, genap = 0
FOR i = 1 to 10
    prompt n
    get n
    IF (n %2 = 0) THEN
        genap = n + genap
    ENDIF
    i = i+1
ENDFOR
display "jumlah = ", genap
END
```

7. Masukkan n_awal dan n_akhir (Bila n_awal > n_akhir, tukarlah keduanya), tampilkan total bilangan ganjil dari n_awal hingga n_akhir.

→ c) program_ketujuh

```
declare integer awal, akhir, i, ganjil, n, s
prompt awal, akhir
get awal, akhir
IF (awal > akhir) THEN
    s = awal
    awal = akhir
    akhir = s
ENDIF
set n=0
FOR i = awal to akhir
    prompt n.
```

No.

Date:

```
get n
IF (n %2 = 1) THEN
    ganjil = n + ganjil
ENDIF
i = i+1
ENDFOR
display "jumlah = ", ganjil
END
```

8. Buatlah pseudocode untuk menghitung berat badan ideal seseorang jika diketahui tinggi badan orang tersebut. Alur tahapan program adalah sebagai berikut :

- Tampilkan teks "Berat Badan Ideal", lalu turun 2 baris
- Tampilkan teks "Tinggi Badan = ", biarkan user mengisi bilangan, kali ini boleh bilangan pecahan (Misalnya 1.90). Tinggi badan dimasukkan dalam satuan meter, tapi dalam perhitungan digunakan centimeter.
- Dari tinggi badan bisa dihitung berat badan ideal adalah "tinggi - 100 - (10% * (tinggi badan - 100))"
  Tampilkan teks "Berat Badan = ", hasil perhitungan berat badan ideal, lalu tampilkan teks "kg"

c)

```
program_kedelapan
    declare float tm, tcm, bb
    display "Berat Badan Ideal"
    endline
    endline
    display "Tinggi Badan ="
    prompt tm
      get tm
    tcm = tm * 100
```

No.

Date:

```
bb = tcm-100-(0.1(tcm-100))
display "Berat Badan = ", bb, "kg"
END
```

9. Buatlah pseudocode untuk menghitung volume sebuah ruang jika diketahui tinggi, lebar, dan panjang ruang tersebut. Alur tahapan program adalah sebagai berikut :

- Tampilkan teks "MENGHITUNG VOLUME RUANG", lalu turun 2 baris
- Tampilkan teks "Panjang (meter)?", biarkan user mengisi bilangan bulat
- Tampilkan teks "Lebar (meter)?", tunggu user mengisi bilangan bulat
- Tampilkan teks "Tinggi (meter)?", sekali lagi user mengisi bilangan bulat.
- Hitung volume ruang
- Tampilkan teks "Volume = ", hasil perhitungan volume, lalu tampilkan teks "(meter kubik)"

c)

```
program_kesembilan
    declare integer p, l, t, vol
    display "MENGHITUNG VOLUME RUANG"
    endline
    endline
    display "Panjang (meter)?"
    prompt p
    get p
    endline
    display "Lebar (meter)?"
    prompt l
    get l
```

No.

Date:

```
endline
display "Tinggi (meter)?"
prompt t
  get t
endline
endline
vol = p * l * t
display "Volume = ", vol, "(meter kubik)"
END
```

10. 1) Buatlah pseudocode untuk kasus dibawah ini
2) Keterangan programnya adalah sebagai berikut
3) Program digunakan untuk menampilkan 5 buah bilangan dalam deret, jika diketahui awal deret dan increment (Pertambahan antar elemen).

c) program kesepuluh

```
declare integer awal, increment, i
display "Menampilkan 5 buah bilangan dalam deret"
endline
endline
display "Awal ? "
prompt awal
  get awal
endline
display "increment ? "
prompt increment
  get increment
endline
display "Bilangan : "
endline
```

No.

Date:

```
endline
set i=1
FOR i = 1 to 5
    display awal, " "
    awal = awal + increment
    i=i+1
ENDFOR
END
```

No.

Date:

1. Buat pseudocode yang menerima input sebuah bilangan bulat dan menampilkan tulisan "bilangan = ..."

```
program_utama
    declare integer x
    prompt x
    get x
    display "bilangan = ...", x
END
```

2. Buat pseudocode yang menerima input 2 bilangan bulat positif, lalu jumlahkan kedua bilangan, terus tampilkan hasilnya.

```
program_kedua
    declare integer a, b, jlh
    DO
        prompt a
        get a
    WHILE (a<0)
    DO
        prompt b
        get b
    WHILE (b<0)
    jlh = a+b
    display "hasil = ", jlh
END
```

3. Buat pseudocode yang menerima input 2 bilangan bulat (Tidak perlu validasi). Tukar kedua bilangan tersebut dan tampilkan.

No.

Date:

c) program_ketiga
declare integer a,b,x
prompt a
get a
prompt b
get b
x = a
a = b
b = x
display a,b
END

4. Buat pseudocode untuk menampilkan bilangan dari bil1 ke bil2
Contoh tampilan : Bil1 = 3
Bil2 = 10
3 4 5 6 7 8 9 10

c) program_keempat
declare integer a,b,i
display "Bil 1 = "
prompt a
get a
display "Bil2 = "
DO
prompt b
get b
WHILE (a>b)
set i=a
FOR i = a to b
display i, " "
i=i+1
ENDFOR
END

No.

Date:

5. Buat pseudo code untuk menampilkan bilangan dari bil 1 ke bil 2 (Tidak perlu validasi). Hanya boleh menggunakan perulangan while.
Contoh tampilan : Bil 1 = 3
Bil 2 = 10
3 4 5 6 7 8 9 10

→ a) program_kelima
declare integer a, b, i
display "Bil1 = "
prompt a
get a
display "Bil2 = "
DO
prompt b
get b
WHILE (a>b)
set i=a
WHILE (i <= b)
display i
i = i+1
display " "
ENDWHILE
END

No.

Date: Week 10

Integer → %d : Bilangan bulat
Float → %f : Pecahan / Real
Character → %c : Karakter

Week 11

In C language contains case sensitive which means uppercase and lowercase letters are different (A≠a).
Keywords / Reserved words in C language are :

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	size of	valatile
do	if	static	while

Data Type	Keyword	Size (Bit)
Character	char / unsigned char	8
Short integer	short / unsigned short	16
Integer	int	16/32
	unsigned int	16
Long integer	long / unsigned long	32
Long long integer	long long / unsigned long long	64
Single-precision floating point	float	32
Double - precision floating point	double	64
Extended - precision floating point	long double	80/96

No.

Date:

- Constants Keywords :
  - # define PI 3.14 → di lvar int main(){
  - const float PI = 3.14; → di dlm int main(){

- a+a=2
  - Operand
  - Operator

- Operators
  - Unary operators → Work on a single operand (a++;)
  - Binary operators → Connect 2 operands (x = a+b)
  - Ternary operators → Has 3 operands (x = a>b ? a-b : b-a;)

- Assign x the value of y : x = y
  - LHS must be a variable ← Associativity (Right to left)
  - RHS must be an expression

- Increment : ++ / a = a+1
- Decrement : -- / a = a-1

- 1 = True
- 0 = False

- Logical
  - AND : x && y
  - OR : x || y
  - NOT : !x

- Bitwise Operators :

1) Bitwise AND : x & y
   - 3 & 5 = 1

```
  011
  101
-----
  001
```

2) Bitwise OR : x | y
   - 3 | 5 = 7

```
  011
  101
-----
  111
```

No.

Date:

3) Bitwise exclusive OR / Gray code : x ^ y

↳ 3 ^ 5 = 6

```
 011
 101
-----
 110
```

4) Bitwise Not / One's complement : ~x

↳ ~3 = -4

```
 0011
-----
(1)100
```

↳ Nilai min

◦ Shift Operators :

1) Shift Left : x<<y = Each bit value in x is moved y positions to the left

↳ 3 << 2 = 12

```
 0 1 1
 0 1 1
0 1 1  → 01100
```

2) Shift Right : x>>y = Each bit value in x is moved y positions to the right

↳ 24 >> 2 = 6

```
 1 1 0 0 0
   1 1 0 0 0
     1 1 0 0 0
```

110

No.

Date:

Compound Assignment Operators :

Operator	Example	Meaning
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y
&=	x&=y	x=x&y
^=	x^=y	x=x^y
|=	x|=y	x=x|y
<<=	x<<=y	x=x<<y
>>=	x>>=y	x=x>>y

Conditional operator : Condition ? Expression 1 : Expression 2

Hint If-Else

- If the result isn't equal to 0 (Condition true), then only expression 1 is evaluated.
- If the result does yield 0 (Condition false), then only expression 2 is evaluated.

Precedence	Operators		Associativity
1	Postfix operators	: ++ --	L to R
2	Unary operators	: ++ -- ! ~	R to L
3	Multiplicative operators	: * / %	L to R
4	Additive operators	: + -	L to R
5	Shift operators	: << >>	L to R
6	Relational operators	: < <= > >=	L to R
7	Equality operators	: == !=	L to R
8	Bitwise AND	: &	L to R
9	Bitwise exclusive OR	: ^	L to R
10	Bitwise OR	: |	L to R

by BOSS

No.

Date:

11	Logical AND : &&		L to R
12	Logical OR : ||		L to R
13	Conditional operator : ? :		R to L
14	Assignment operators : = += -= *=		R to L
		/= %= ^= |=	
		&= <<= >>=	

- Example : a/b %c ≡ (a/b) %c ← L to R
  a=b=c ≡ a=(b=c) ← R to L

- Postfix operators : x++ → Precedence number 1
- Prefix operators : ++x

- scanf ("%d", &number) : Contoh input di bahasa C
  - "%d" : Identify
  - scanf : Obtain value from user (Read from standard input which is usually the keyboard)
  - int : Variable
  - &number : Ampersand (Address operator)

- The ampersand, when combined with the variable name, tells scanf the location / address in memory at which the variable is stored.

Week 12

- char nama [100]
  - 100 : Nilai batas karakter
- scanf ("%[^\n]", &nama) = gets (nama)
  - "%[^\n]" : Bisa input bbrp kata
- puts (nama) = Output lgsg bbrp kata

No.

Date:

- scanf ("%[^t]", &nama)
  input sampai huruf t, maka proses lgsg ljt
- scanf ("%s", &nama)
  hanya bs 1 kata / sampai spasi
- getchar → a = getchar() → output a hrs dgn enter (sama aja dgn scanf)
- getche → a = getche() → output q lgsg keluar dgn inputnya
- getch → a = getch() → output a lgsg kluar kl tkn keyboard

- abs(x) → Nilai x mutlak
  (absolute)

- b = scanf ("%d",&a);
  ↳ Output b adalah banyak inputan berarti 1
  b = scanf ("%d %d", &a, &d);
  ↳ Output b adalah banyak inputan berarti 2

- d = printf ("b = %d\n", b)
  ↳ Misalkan b=2, maka output d adalah 6, karena dihitung karakternya yaitu b, spasi, =, spasi, %d, dan \n.
  ↳ Jika keluarannya adalah b = 1234, maka output d adalah banyaknya karakter dari b yaitu b, spasi, =, spasi, 1, 2, 3, 4, \n, maka d = 9
  ↳ Kalau tidak \n, maka d = 8

- fflush(stdin) berfungsi utk membersihkan buffer keyboard utk menghindari inputan berupa spasi / enter.

- %c utk huruf
- %s utk kata / lbh

a = 10
i = 5

i++
a += i++ + 1

No.
Date:

- getchar, getche, getch, putch, putchar pakai %c
- gets, puts pakai %s

- %5d berarti 5digit ke kanan → ___57
- %-5d berarti 5 digit kekanan → 57___

Header <stdio.h> yang memanggil fungsi gets, getchar, getche, getch, puts, putchar, putch. Function prototype tells the compiler :

- The type of data returned by the function
- The number of parameters the function expects to receive
- The types of the parameters
- The order in which these parameters are expected.

Function prototype :

```
return_value_type function_name(parameter_list);
```

- function_name is any valid identifier, such as rand, abs, pow, clrscr, etc.
- return_value_type is the data type of the result returned to the caller, such as void that indicates a function doesn't return a value. Other examples are int, double, etc.
- parameter_list is a comma-seperated list that specifies the parameters received by the function when it's called. If a function doesn't recieve any value, parameter_list is void. A type must be listed explicitly for each parameter. Ex : (int x), (void), (double x, double y)

No.

Date:

Precise output formatting is accomplished with printf:

printf(format_control_string, other_arguments);

- format_control_string describes the output format. other_arguments is optional, corresponds to each conversion specification in format control string
- Each conversion begins with % and ends with a conversion specifier, and can be many conversion specification in 1 format control string.

Conversion Specifications:

Output

% [flags] [field_width] [.precision] [length modifier] specifier

- The parts of this syntax that are indicated in square brackets are all optional, but the order must followed.
- Any conversion specification can include a field width.
- The precision doesn't apply to all conversion types.

- %d = 95
  %5d = ___95
  %-5d = 95___
  %05d = 00095
  % d = _95
  %+d = +95
- %f = 3.141593
  %.3f = 3.142
  %07.2f = 0003.14

No.

Date:

Output
Conversion specifiers

Conversion Specifier	Description
d	Signed decimal integer
i	Signed decimal integer
o	Unsigned octal integer
u	Unsigned decimal integer
x or X	Unsigned hexadecimal integer → X causes the digits 0-9 and the letters A-F to be displayed → x causes the digits 0-9 and the letters a-f to be displayed
e or E	Signed floating-point number in exponential notation. → E and e print exactly 1 digit to the left of the decimal point → E prints uppercase E preceding the exponent → e prints lowercase e preceding the exponent
f	Signed floating-point number in fixed-point notation
c	Character
s	String

Output
Flags

Flag	Description
-	Left justify the output within the specified field
+	Display + preceding positive values and - negative values
space	Print a space before a positive value not printed w/ the + flag
0	Pad a field with leading zeros
#	Prefix 0 to the output value when used with the octal conversion specifier o

	Prefix 0x or 0X to the output value when used with the hexadecimal conversion specifiers x or X.
	Force a decimal point for a floating-point number printed with e, E, or f that doesn't contain a fractional part (Normally the decimal point is printed only if a digit follows it).

- Output Length Modifiers :

Length Modifier	Description
h	Indicates that a short integer is displayed
l	Indicates that a long integer is displayed
L	Indicates that a long double floating-point value is displayed.

- %d = 95
  %o = 137
  %X = 5F
  %#x = 0x5f

Input Conversion specifications :

[% [*] [field_width] [length_modifier] specifier]

- * is an assignment suppression character
- The assignment suppression character enables scanf to read any type of data from the input and discard it without assigning it to a variable.

No.

Date:

Input

- Conversion Specifiers :

Conversion specifier	Description
d	Signed decimal integer
i	Signed decimal, octal, or hexadecimal integer
o	Octal integer
u	Unsigned decimal integer
x or X	Hexadecimal integer
e, E, or f	Floating-point value
c	Character
s	String

Input

- Length Modifiers :

Length Modifier	Description
h	Indicates that a short integer is to be input
l (With integer)	Indicates that a long integer is to be input
l (With floating-point numbers)	Indicates that a double value is to be input
L	Indicates that a long double value is to be input

- Precise input formatting can be accomplished with scanf :

```
scanf ( format_control_string, other_arguments );
```

- format_control_string describes the formats of the input
- other_arguments are pointers to variables in which the input will be stored.

- Conditional bernilai true selain 0, {-1, -2, -3, 10} true
- == berarti penentuan kondisi
  = berarti kesh. nilai

No.

Date: Week 14

The break statement, when executed in while, do...while, for, or switch statement, causes an immediate exit from the statement. The continue statement, when executed in while, do...while, or for statement, skips the remaining statements in the body of that control statement and performs the next iteration of the loop. In while and do...while, the loop repetition is evaluated immediately after the continue statement is executed. In for, the update expression is executed, then the loop repetition is evaluated.

Week 15

Best way to maintain large program is to construct it from smaller pieces or modules. This technique is called divide and conquer. Reasons are for more modularity, readibility, and code reusability. Modules in C = functions.

- C Standard Library
  - ↳ Prepackaged functions
  - ↳ Provides a rich collection of functions for performing common math, string manipulations, character manipulations, input/output, etc.

- Programmer-defined Functions
  - ↳ Programmer can write functions to define specific tasks that may be used at many points in a program.

Each standard library has a corresponding header containing the function prototypes for all the functions in that library and definitions of various data types and constants needed by those functions.

No.

Date:

Standard Library Header	Explanation
<stdio.h>	Standard input / output library functions
<math.h>	Math library functions
<string.h>	String - processing functions
<time.h>	Time and date manipulation functions
<stdlib.h>	Conversions of numbers to text and text to numbers, memory allocation, random numbers, and other utility functions.
<ctype.h>	Functions that test characters for certain properties Functions that can be used to convert lowercase letters to uppercase letters and vice versa

Programmer - defined functions

Merupakan funtion definition

```
return_value_type function_name (parameter_list) {
  statement;}
```
⇒ Refered as function header

- function_name is any valid identifier
- return_value_type is the data type of the result returned to the caller.
- parameter_list is a comma-seperated list that specifies the parameters

Merupakan function prototype

```
return_value_type function_name (parameter_list);
```

- The compiler uses function prototypes to validate function calls
- A function call that doesn't match the function prototype is a syntax error.

No.

Date:

- An error is also generated if the function prototype and the function definition disagree.
- A function prototype tells the compiler :
  - ↳ The type of data returned by the function
  - ↳ The number of parameters the functions expects to receive.
  - ↳ The types of the parameters.
  - ↳ The order in which these parameters are expected

Call-by-Value Functions : Hrs ada return a variable

↳ When arguments are passed by value, a copy of the argument's value is made and passed to the called function.

↳ Changes to the copy do not affect an original variable's value in the caller.

↳ In C, all calls are by value

Call-by-Reference Functions : Gush ada return a variable kan pakai pointer

↳ When an argument is passed by referene, the caller allows the called function to modify the original variable's value.

A recursion function is a function that calls itself.

No.

Date: Week 16

Pointers are variables whose values are memory addresses. A pointer contains an address of a variable that contains a specific value. A variable name directly references a value, but a pointer indirectly references a value.

```
data_type *pointer_name;
```

Pointers should be initialized either when they are defined or in an assignment statement. A pointer may be initialized to NULL or an address. A pointer with the value NULL points to nothing.

```
int number = 5;
int *numberPtr;
numberPtr = &number;
printf("%d", *numberPtr);
```

The address operator/ampersand (&) returns the address of its operand /variable. The indirection operator/ dereferencing operator (*) returns the value of the object to which its operand (Pointer) points.

* & saling berkomplemen.

Array is a group of memory locations with same name and same type. First element starts with 0 index. The position number contained within square brackets is more formally called a subscript or index that must be an integer or integer expression.

No.

Date:

element_data_type array_name [size];

array_name [Index];

If there are fewer initializers than elements in the array, the remaining elements are initialized to 0 (int a[100] = {0};). This explicitly initializes the first element to 0 and initializes the remaining 99 elements to 0.

If the array size is omitted from a definition with an initializer list, the number of elements in the array will be the number of elements in the initializer list. (int b[] = {37, 19, 85, 46, 93};). This would create a 5-element array.