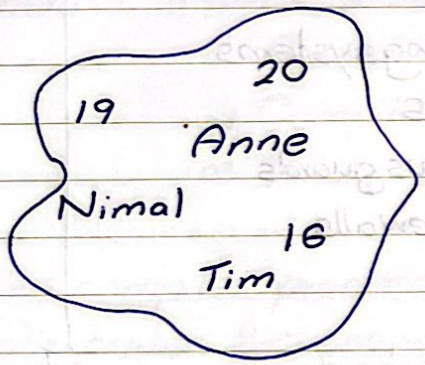
Date 30/05/2022

What is a Computer?

* A computer is an electronic device which is used to convert data into information.
* Data are raw facts. In computing these data transformed into information.
* In other words information are processed facts. (data)
* Converting data into information is called as data processing.

Data Processing



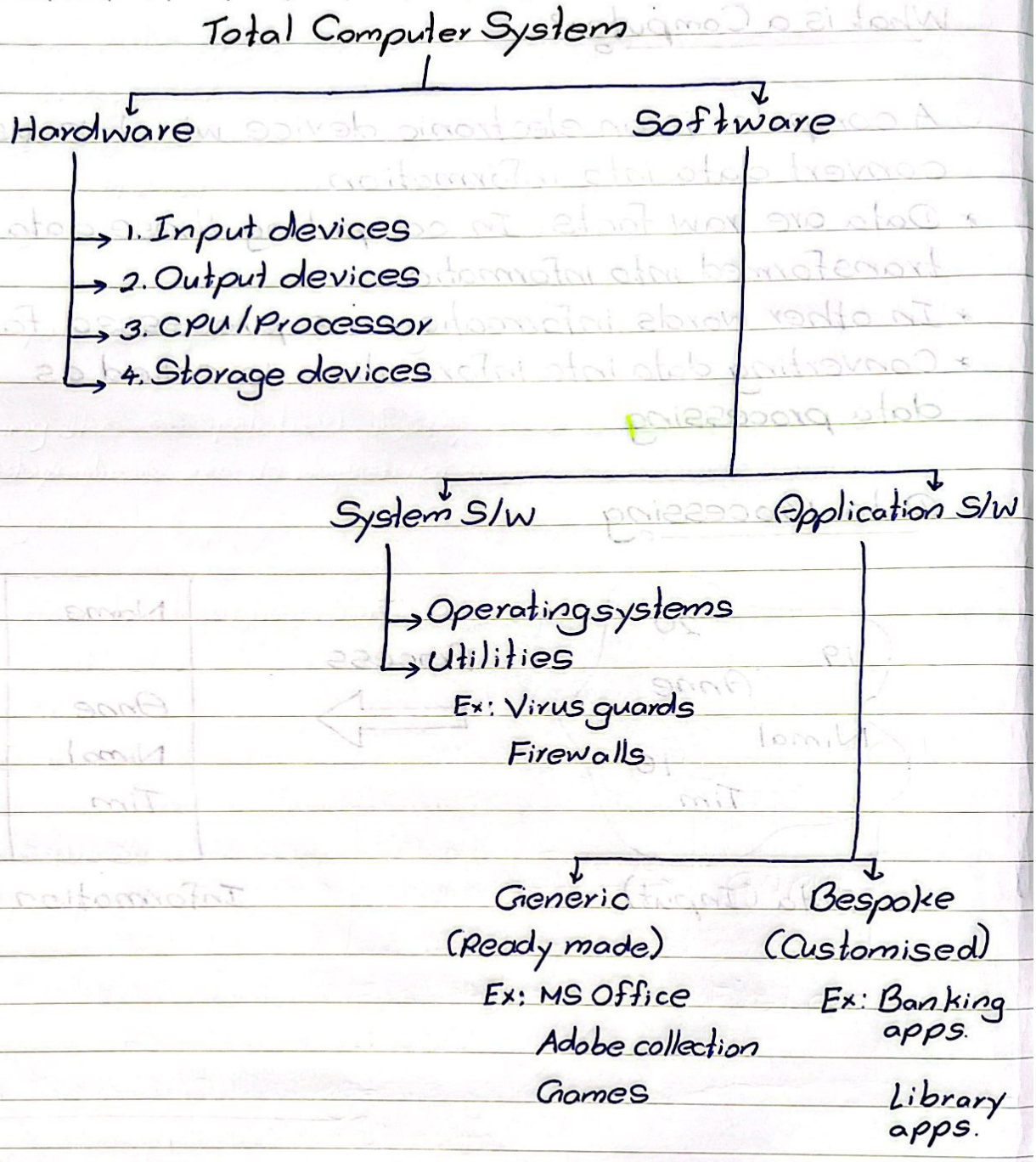
Data (Input)

Process

Name	Age
Anne	16
Nimal	19
Tim	20

Information (Output)

Date No



Software

* Simply softwares are collection of programs.
* A program is a set of instructions given to the computer by using a programming language.

Date No

* Programming languages are broadly divided into 3 catergories.

low level { 01. Machine language. (Binaries - 0, 1)
02. Essembly language. (Set of symbols)
03. High level language. (Similar to natural English statements)
Ex: C, Java, Python, PHP

Language Translators.

06/06/2022

* The purpose of programming language translator is to convert the code written by programmer into machine code.

* There're 3 types of language translators in use.
01. Assembler - use assembly language.
02. Compiler } High level language.
03. Interpreter }

* The compiler process the whole set of lines at once. But in interpreter the translation happens line by line.

C Programming

First Program in C language - Hello World Program.

01. Install a C editor such as Code Blocks, Turbo C or any other C compatible editor.

02. Type the following C source code and save the file.

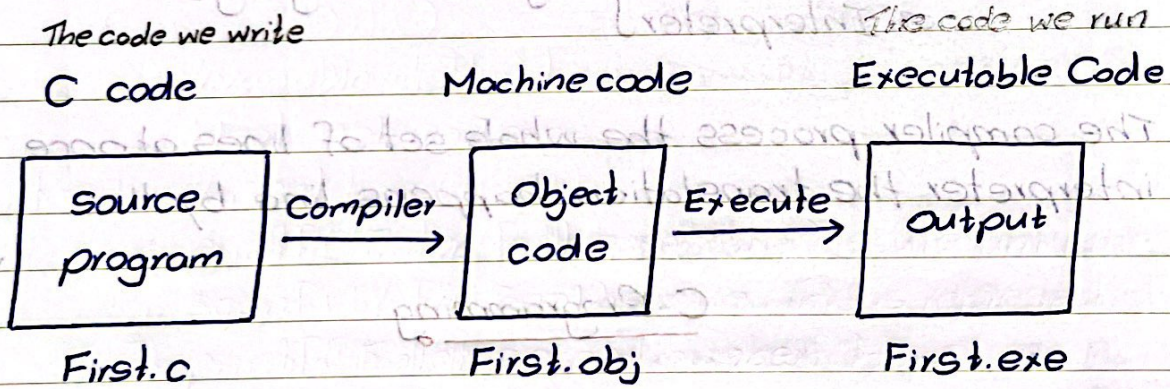
```
#include<stdio.h>
int main ()
{
    Printf("Hello World!");
}
```

for every code

03. Compile the program using the editor.

04. During the compilation if any error exist it will be displayed. Correct the errors and re-compile the program.

05. To view the output, run the program.



Data Input in C language.

* In programming the input is associated with variables.
* A variable is a temporary memory location.
* You can store different types of values inside a variable.
* Prior to using variables, it must be declared.

*variable is a temporary memory location.

Date No

Variable Declaration.

* In programming variables are declared with the data type.

```
int    no1, no2
```

int → Data type; no1, no2 → variable names

(01.) Write a C Program to input two numbers and display the total.

```
#include<stdio.h>
int main()
{
  //variable declaration.          //→comments
  int no1, no2, total;
  //data input
  printf("Enter first number ");    *add '&' before
  scanf("%d", &no1);                the variable in
  printf("Enter second number ");   scanf.
  scanf("%d", &no2);
  //process
  total = (no1 + no2);
  //output
  printf("The total is %d", total);
}
```

Data Types

Data type	Conversion specifier (Input)
integers int	%d
fractions float	%f
characters char (Single character)	%c %s (more than 1 character)

(02.) Write a program to input 2 numbers with fractional values and display the average value.

```
#include<stdio.h>
int main()
{
    //variable declaration.
    Float no1, no2, avg;
    //input
    printf("Enter first number ");
    scanf("%f", &no1);
    printf("Enter second number ");
    scanf("%f", &no2);
    //calculate the average.
    avg = (no1 + no2) / 2;
    //display the average
    printf("The average is %.2f", avg);
}
```

%.2f → දශමස්ථාන දෙකකට

Date 20/06/2022

Working with String

```
{
    //character array ↙ Number of characters
    char name [20];
    printf("Enter your name.");
    scanf("%s", &name);
    printf("Hey %s", name);
}
```

(01.) Write a C Program to allow the user to input name, birth year and display name with age.

```
{
    int byear, age;
    char name [15];
    printf("Enter your name");
    scanf("%s", &name);
    printf("Enter your birth year");
    scanf("%d", & byear);
    age = 2022 - byear;
    printf("Hey %s, you are %d years old", name, age);
}
```

Date No

Control Structures

* In a computer program the flow of program statements is called as control structures.

* There're 3 types of control structures

01. Sequence
    - In here statements are executed line by line in the given order.

02. Selection
    - This flow of statements are controlled by the conditions provided by the following 2 constructs.
        a) if()
        b) switch()

03. Interation / repetition (loops)
    - set of statements will repeatedly execute until the given condition remains true. This is represented by using the following 3 types of loops.
        a) for loop
        b) while loop
        c) do while loop
    - all the loops must begin with the value and there must be a end point. (value)

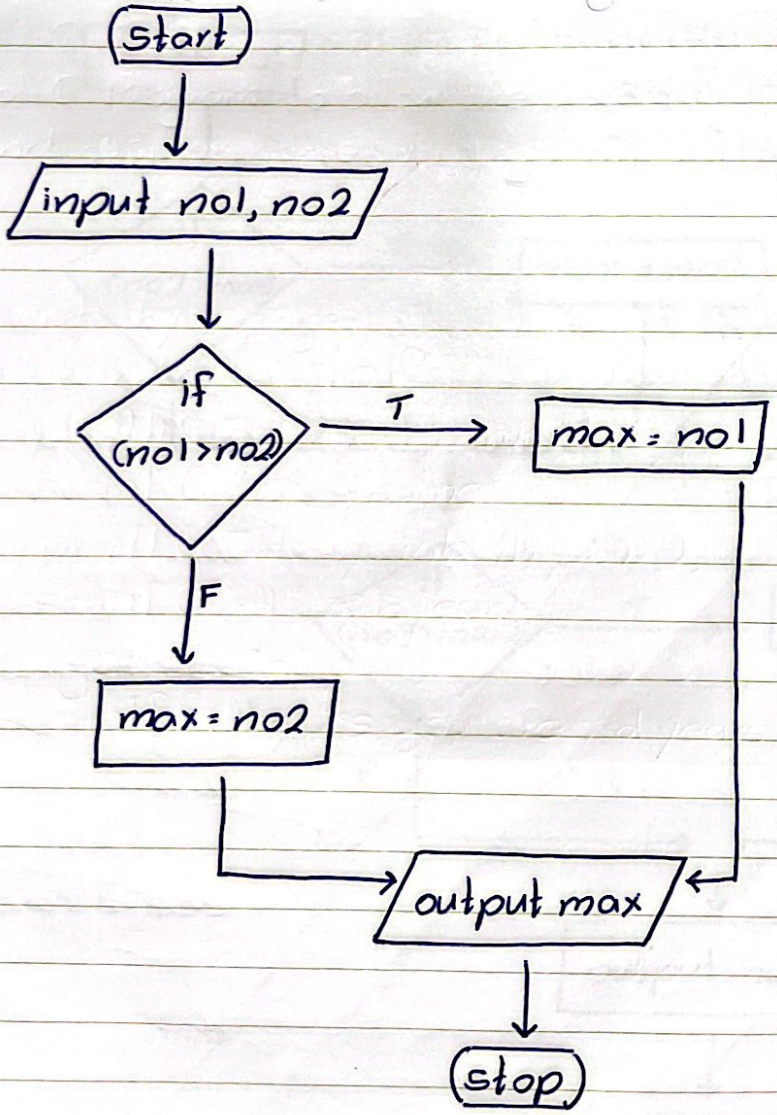
Date No

Selection Structure

a) if () condition

(01.) Write a program to allow the user to input 2 numbers and display the highest number.

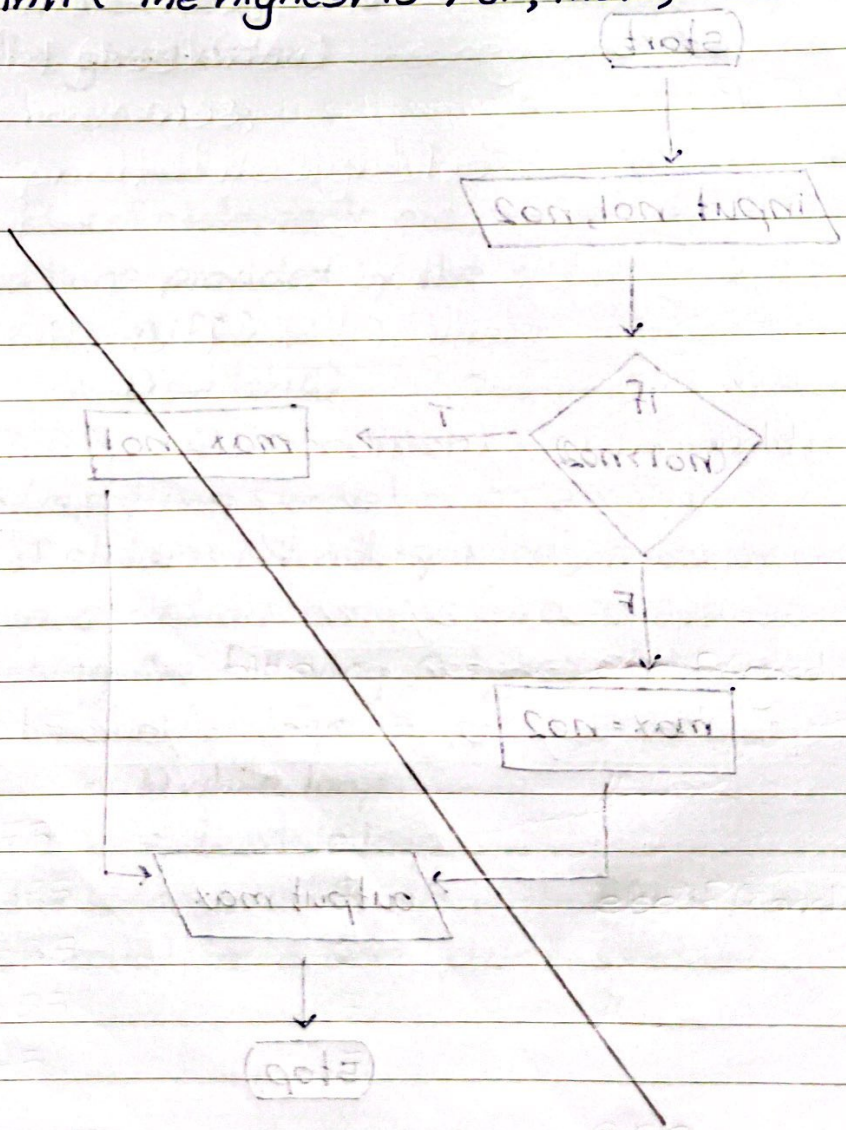
Flow chart



Date No

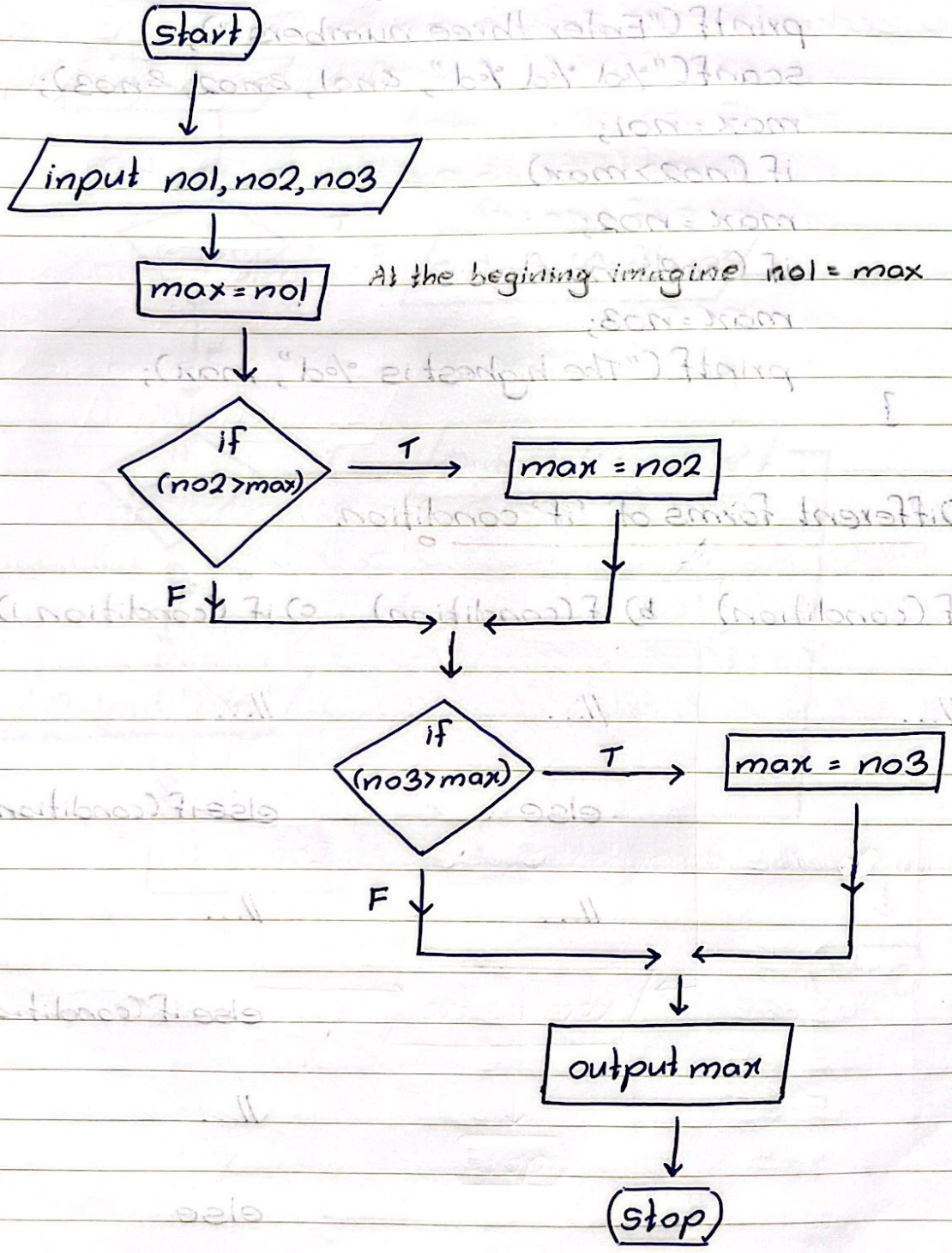
```
{
    int no1, no2, max;
    printf ("Enter two numbers ");
    scanf ("%d %d",    &no1, &no2);
    if (no1>no2)
    max = no1;
    else
    max = no2;
    printf ("The highest is %d", max);
}
```

* don't use semicolon



Date No

(02.) Modify the above program to allow the user to input 3 numbers and display the highest number.



Date No

```
{
    int no1, no2, no3, max;
    printf("Enter three numbers ");
    scanf("%d %d %d", &no1, &no2, &no3);
    max = no1;
    if (no2>max)
    max = no2;
    if (no3>max)
    max = no3;
    printf("The highest is %d", max);
}
```

① Different forms of 'if' condition.

a) if (condition)

```
//...
```

b) if (condition)

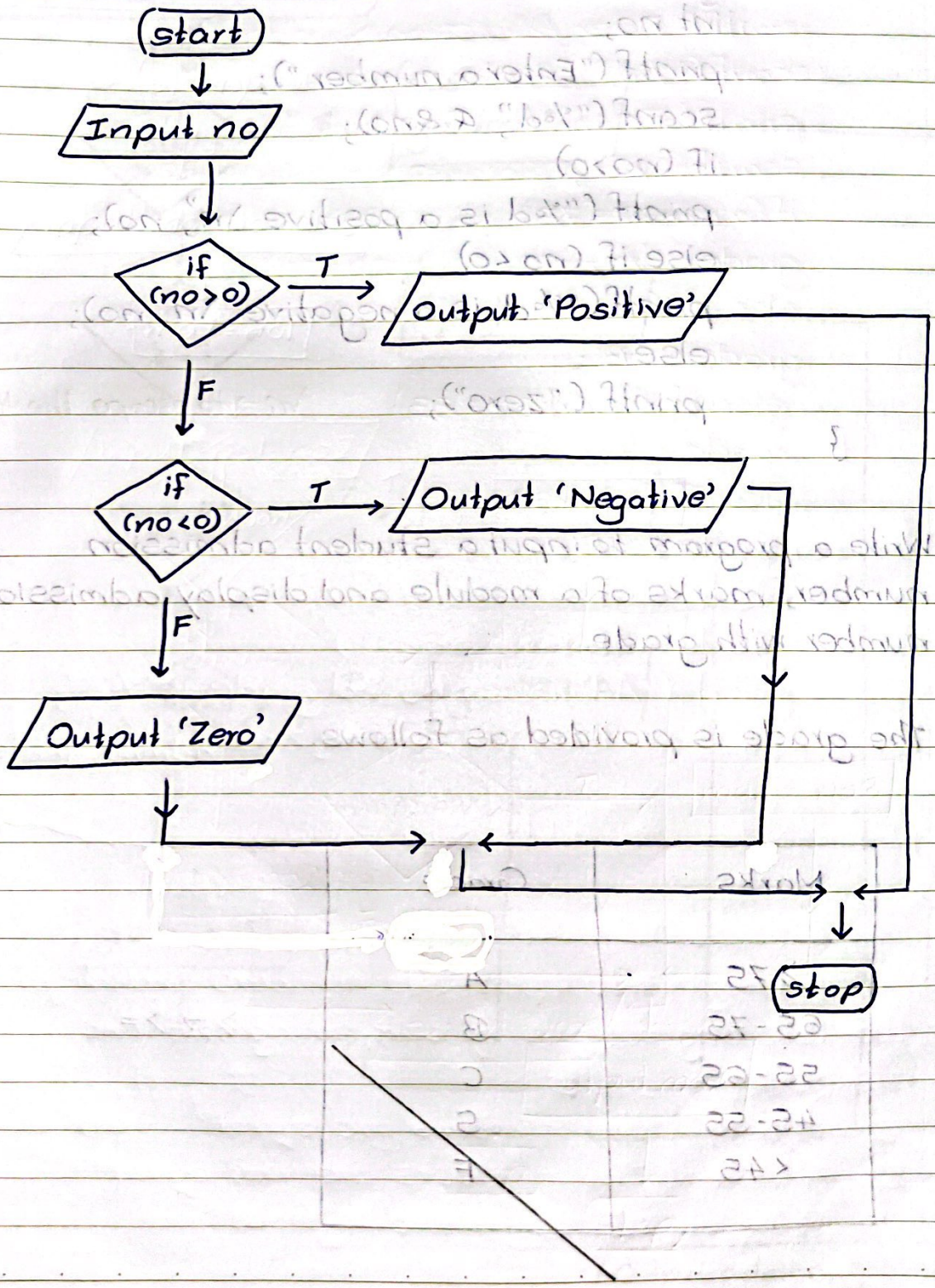
```
//...
else
//...
```

c) if (condition 1)

```
//...
else if (condition 2)
//...
else if (condition 3)
//...
else
//...
```

Date 27/06/2022

(01.) Write a program to input a number and display the entered number is a positive, negative or zero.



```
{
    int no;
    printf("Enter a number ");
    scanf("%d", &no);
    if (no>0)
    printf("%d is a positive \n", no);
    else if (no<0)
    printf("%d is a negative \n", no);
    else
    printf("zero");
}
```

\n → to escape the line

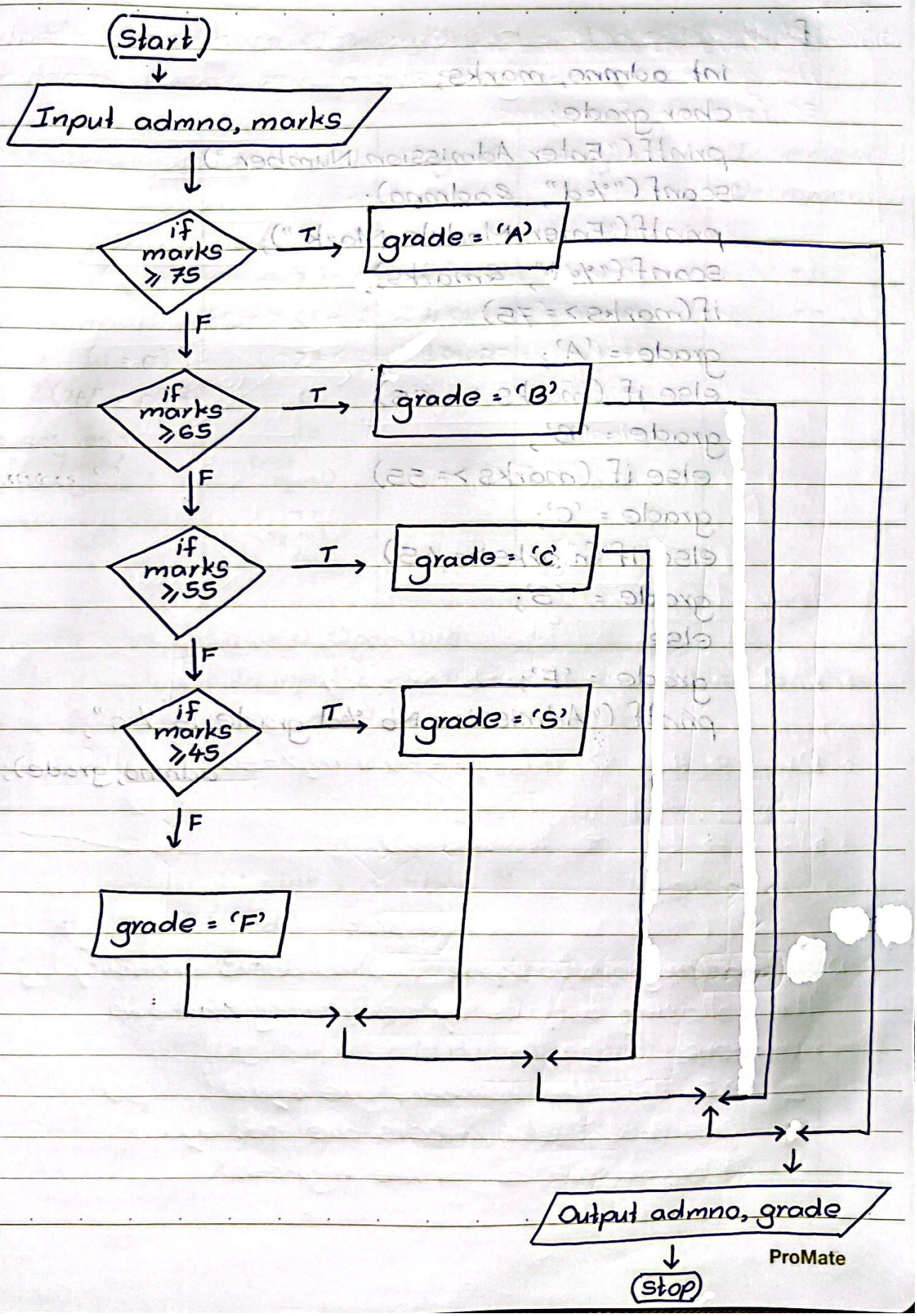
(02.) Write a program to input a student admission number, marks of a module and display admission number with grade.

The grade is provided as Follows.

Marks	Grade
≥75	A
65-75	B
55-65	C
45-55	S
<45	F

≥ → >=

Date No

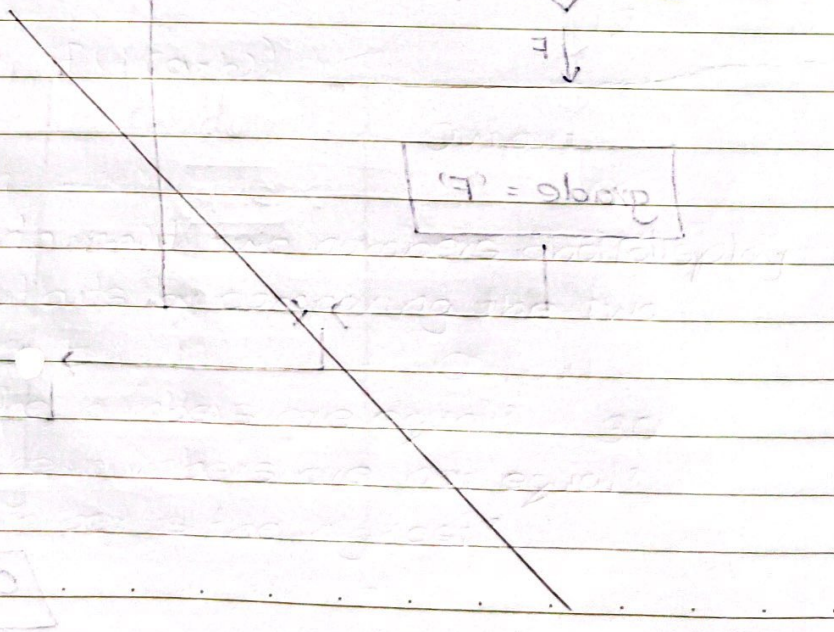


ProMate

Date No

```
{
    int admno, marks;
    char grade;
    printf("Enter Admission Number ");
    scanf("%d", &admno);
    printf("Enter Module Mark ");
    scanf("%d", &marks);
    if(marks >= 75)
    grade = 'A';
    else if (marks >= 65)
    grade = 'B';
    else if (marks >= 55)
    grade = 'C';
    else if (marks >= 45)
    grade = 'S';
    else
    grade = 'F';
    printf ("Admission No %d grade is %c",
                admno, grade);
```

if (a = 1)
if (a = "A")
char දාගෙන්
'' දාගමා



Date No

C Operators

Arithmetic	Relational	Logical	Assignment	Increment/ Decrement
+	>	&& (and)	+=	++
-	<	|| (or)	-=	--
*	>=	! (not)	*=	
/	<=		/=	
% (modulus / remainder)	!= (not equal)			
බෙදුවම එම ඉතිරිය අගය	== (equal to)			

Arithmetic and Relational Operators.

(01) Write a program to input a number, check and display the entered number is an odd or even number.

```
{
    int no;
    printf ("Enter a number ");
    scanf ("%d", &no);
    ans = no%2;  // remainder / modulus
    if (ans == 1) //equal to
    printf ("%d is an odd number", no);
    else
    printf ("%d is an even number", no);
}
```

no 2ක් බෙදුවම ඉතිරියෙන අගය = ans

Date No

Logical Operators.

* And → && → if (a>b && a>c)
* Or → || → if(m1>=50 || m2>=50)
* Not → ! → if(!(a>b))

(01.) Write a program to input marks of two modules and display the grade as pass, if the student has obtained marks above 50 for both of modules.

```
{
    int m1, m2;
    printf("Enter marks of two modules ");
    scanf("%d %d", &m1, &m2);
    if (m1>=50 && m2>=50)
    printf("Pass \n");
    else
    printf("Fail \n");
}
```

(02.) Write a program to input two numbers and display the following outputs by comparing the two numbers.

a) Is that the numbers are equal?
b) Is that the numbers are not equal?
c) Which number is the highest?

Date No

```
{
    int no1, no2;
    printf ("Enter two numbers ");
    scanf ("%d %d", &no1, &no2);
    if (no1 == no2)
    printf ("Numbers are equal \n");
    if(no1! = no2)
    printf ("Numbers are not equal \n");
    if(!(no1>no2))
    printf("Second is greater than first \n");
    else
    printf("First is greater than second \n");
}
```

04/07/2022

Assignment Operators.

(01.) Write the output of the following program.

```
int a=10;
a+=20;          // 10+20
prinf("a=%d \n", a);
a-=5;           // 30-5
printf("a=%d \n", a);
a*=10;          // 25*10
printf("a=%d \n", a);
a/=5;           // 250/5
printf("a=%d \n", a);
```

Output:

a=30
a=25
a=250
a=50

Date No

Increment / Decrement Operators

++ (increment by one)
-- (decrement by one)
x++ (post increment)
++x (pre increment)
x-- (post decrement)
--x (pre decrement)

(01.)
```
int x=10;
x++;
printf("%d \n", x);
++x;
printf("%d \n", x);
```
Output:
11
12

(02.)
```
int x=10;
printf("%d \n", x++);
printf("%d \n", ++x);
```
Output
10
12

(03.)
```
int x=10;
printf("%d \n", --x);
printf("%d \n", x--);
```
Output
9
9

(04.)
```
int x=10, y=20, z;
z = x++ + y++;
printf("%d \n", z);
```
Output
30

Date No

(05) `int x=10, y=20, z;`
`z = ++x + y++;`
`printf("%d \n", z);`

Output:
31

Switch Conditional Structures

* The switch structure is an alternative to the if structure.
* In certain programs using switch you can write the program in less number of statements.

(01.) Write a program to allow the user to input month number and display the name of the month.

* Using if condition

```
int m;
printf("Enter month number ");
scanf("%d", &m);
    if (m==1)
printf("January");
else if(m==2)
printf("February");
else if(m==3)
printf("March");
else if(m==4)
printf("April");
else if(m==5)
printf("May");
```

```
else if (m==6)
printf ("June");
else if (m==7)
printf("July");
else if (m==8)
printf("August");
else if (m==9)
printf ("September");
else if (m==10)
printf("October");
else if (m==11)
printf("November");
else if (m==12)
printf("December");
else
print f ("Invalid Month Number");
```

* Using switch condition

```
int m;
printf("Enter Month Number ");
scanf ("%d", &m);

switch (m)
{
    case 1: printf ("January"); break;
    case2 : printf ("February"); break;
    case 3 : printf("March"); break;
    case 4 : printf("April"); break;
    case 5 : printf("May"); break;
```

End the line

```
case 6 : printf("June"); break;
case 7 : printf("July"); break;
case 8 : printf("August"); break;
case 9 : printf("September"); break;
case 10 : printf("October"); break;
case 11 : printf("November"); break;
case 12 : printf("December"); break;
default : printf("Invalid Month Number");
```

(02) Write a program using switch structure to allow the user to input a character that check and display the entered character is a vowel or not an vowel.

```
char v;
printf("Enter a character");
scanf("%c", &v);
switch (v)
    {
      case 'a' : printf("a is a vowel"); break;
      case 'e' : printf("e is a vowel"); break;
      case 'i' : printf("i is a vowel"); break;
      case 'o' : printf("o is a vowel"); break;
      case 'u' : printf("u is a vowel"); break;
      default : printf("%c is not a vowel", v); break;
```

Date 11/07/2022

Iteration / Repition Control Structure (Loops)

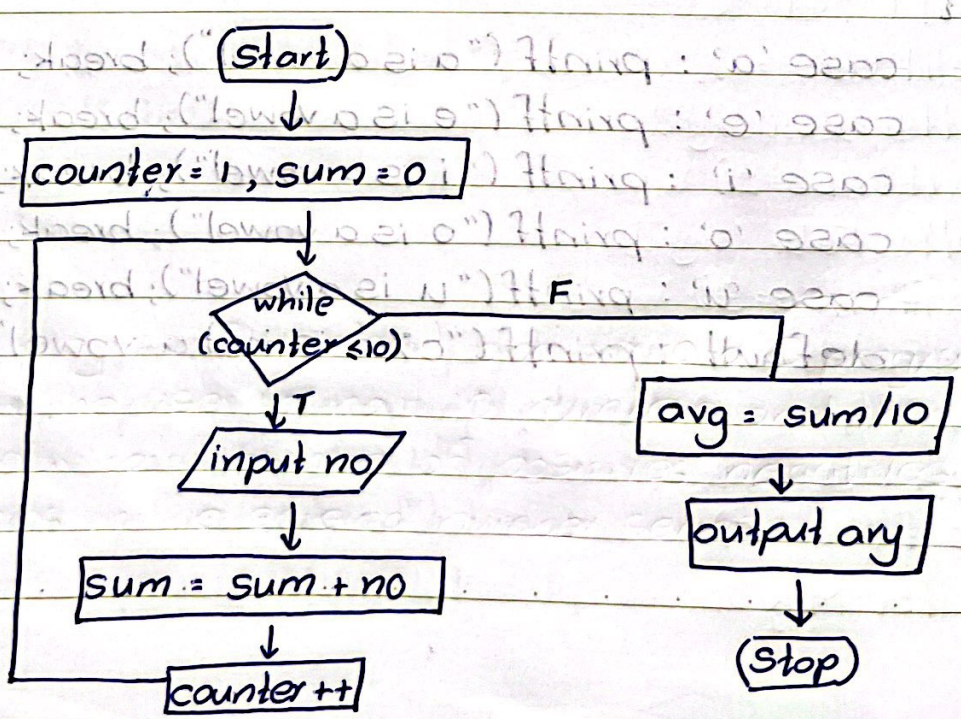
* In this structure the given statements repeatedly execute until the given condition remains true.
* all the loops must begin with the value and there must be a end point (value).

While loop

Syntax:

```
while (condition)
{
    //body
}
```

(01.) Draw a program flow chart and write a C program to allow the user to input 10 integers and then to display the average value.



Date No

```
int counter = 1, sum = 0, no;
float avg;
while (counter <= 10)
{
    printf ("Enter %d number ", counter);
    scanf ("%d", &no);
    sum = sum + no;
  counter++;  ← add 1 to the counter
}
```

casting → convert to decimal

```
avg = (float) sum/10;
printf ("The average is %2.f \n", avg);
```

(02.) Using while loops display the following outputs.

a) 1 2 3 ... 100

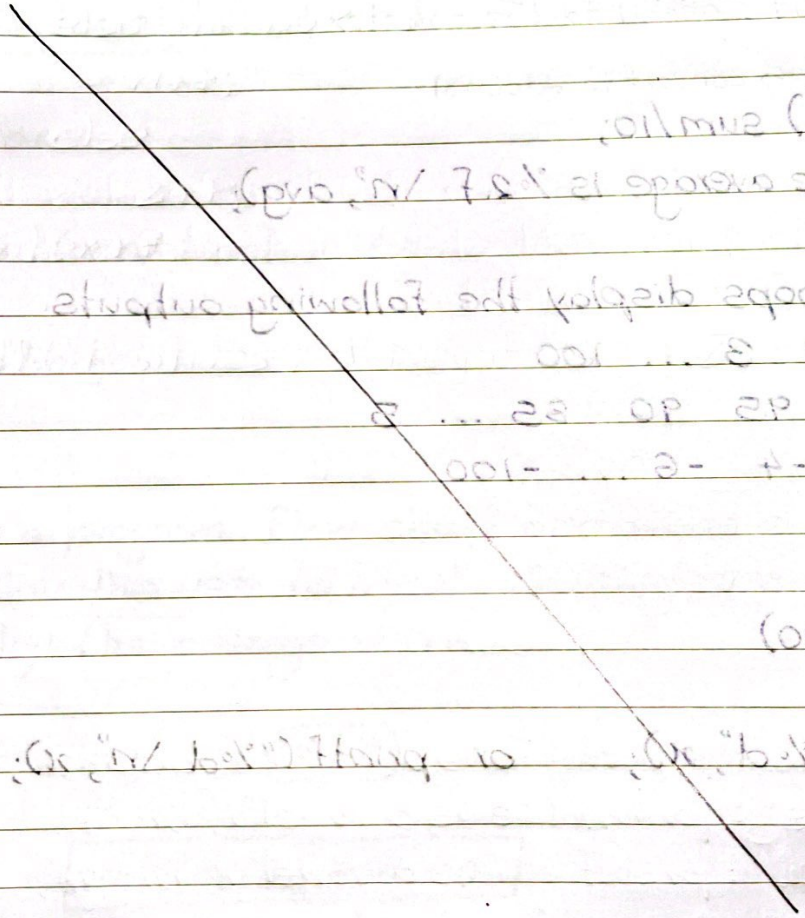
b) 100 95 90 85 ... 5

c) -2 -4 -6 ... -100

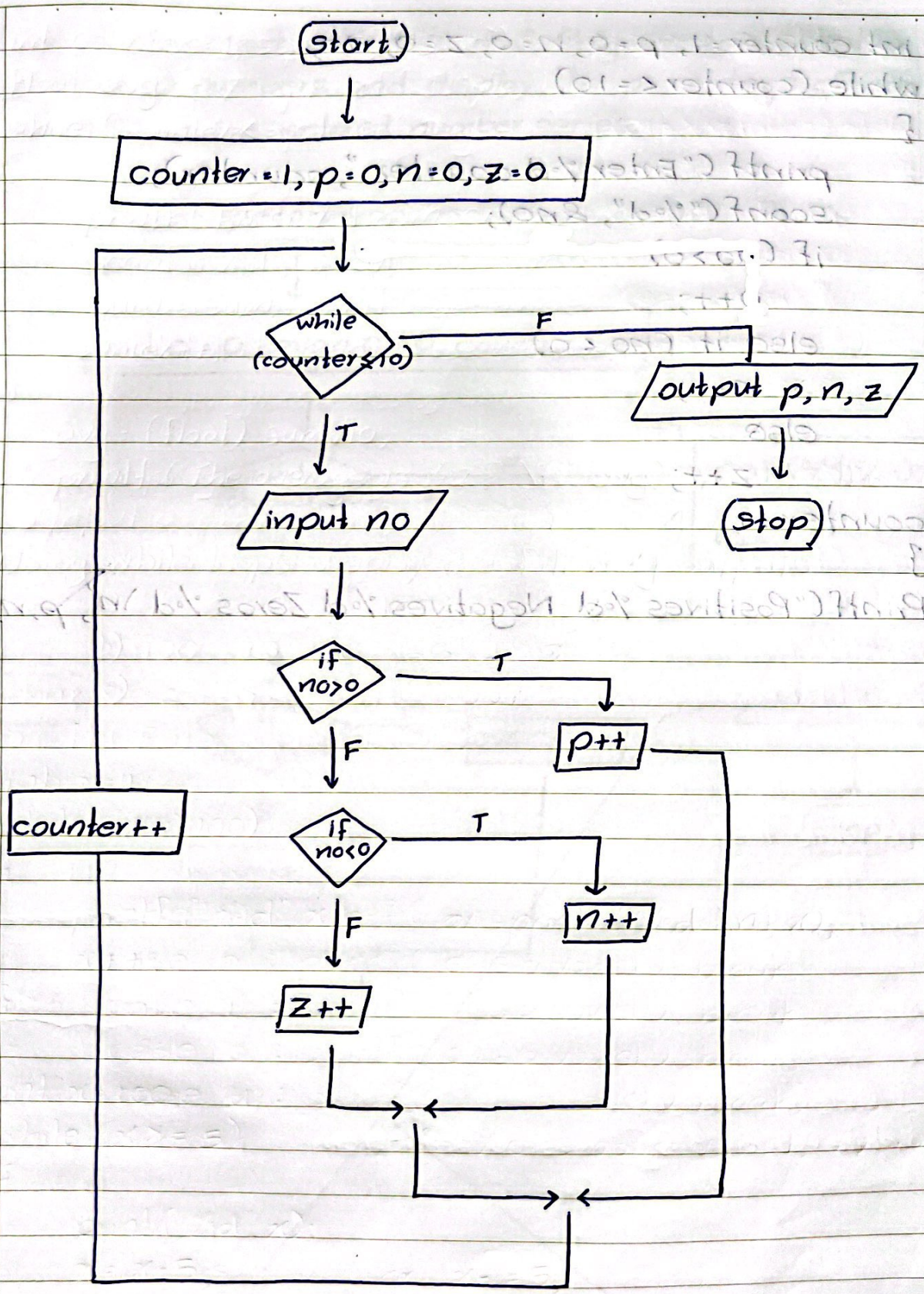
a)
```
int x = 1;
while (x <= 100)
{
    printf ("%d", x);     or printf ("%d \n", x);
    x++;
}
```

b)
```
int x = 100;
while (x >= 5)
{
    print ("%d", x);
    x = x - 5;     or x -= 5;
```

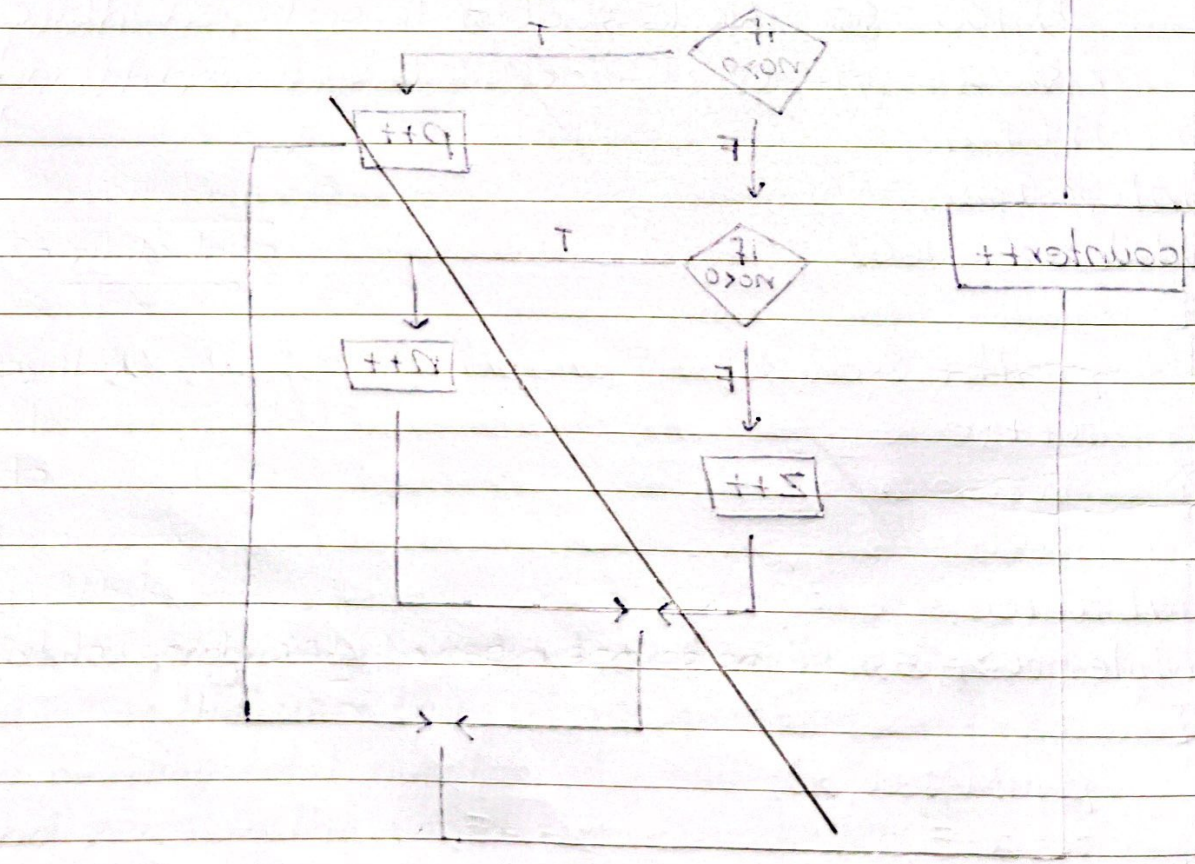
c)
```
int x = -2;
while (x >= -100)
{
    printf ("%d", x);
    x = x - 2;
}
```



(03.) Drow a program flow chart and write a C program to allow the user to input 10 numbers and then to display total number of positives, negatives and zeros in the entered number series.

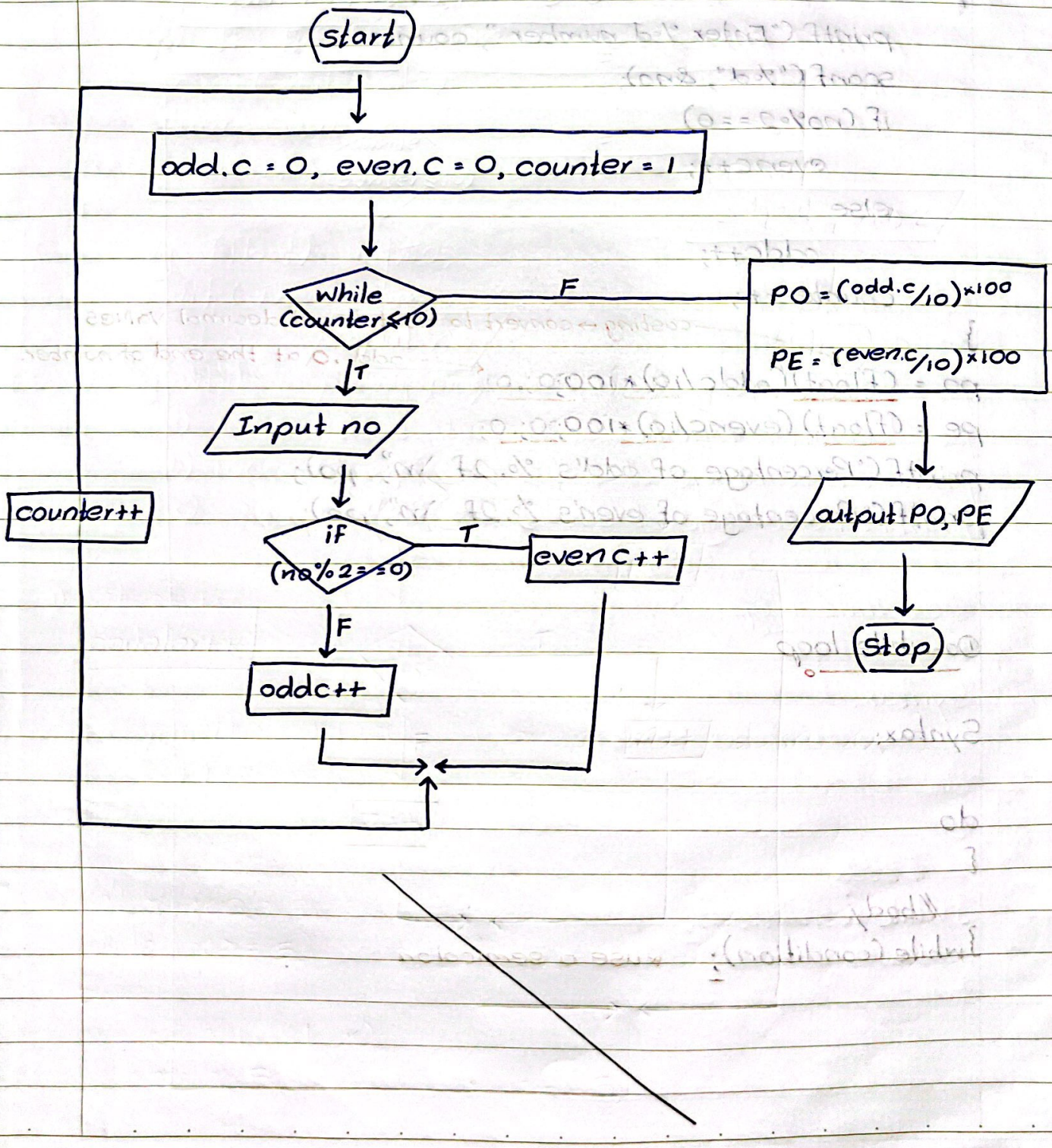


```
int counter = 1, p = 0, n = 0, z = 0, no;
while (counter <= 10)
{
    printf ("Enter %d counter", counter);
    scanf ("%d", &no);
    if (no > 0)
        p++;
    else if (no < 0)
        n++;
    else
        z++;
counter++;
}
Printf ("Positives %d Negatives %d Zeros %d \n", p, n, z)
```



Date 18/07/2022

(01.) Draw a Flow chart and write a C Program to allow the user to input 10 numbers and display the percentage of odd, even numbers in the entered number series.



```
int oddc = 0, evenc = 0, no, counter = 1;
float po, pe;
while (counter <= 10)
{
    printf ("Enter %d number ", counter);
    scanf("%d", &no);
    if (no%2==0)
        evenc++;
    else
        oddc++;
    counter++;
}
po = (float)(oddc/10.0) * 100.0;
pe = (float)(evenc/10.0) * 100.0;
printf("Percentage of odd's %.2f \n", po);
printf("Percentage of even's %.2f \n", pe);
```

casting → convert to float

add .0 at the end

① Do-While loop.

Syntax:

```
do
{
  //body
}while (condition);
```

* use a semicolon

Date No

(01.) By using do-while loops, display the following outputs.

a) 1 2 3 ... 100

b) 10 20 30 ... 1000

c) 100 99 98 ... 1

a)
```
int x=1;
do
{
  printf("%d \n", x);
  x++;
} while (x<=100);
```

b)
```
int x=10;
do
{
  printf("%d \n", x);
  x=x+10;
} while (x<=1000);
```

c)
```
int x=100;
do
{
  printf("%d \n", x);
  x--;
} while (x>=1);
```

Date No

* In the while loop, condition is provided at the begining of the loop. It means the body of the while loop executes only if the condition is true.
* But in do-while loop, condition is provided at the end of the loop. Therefore the body will execute atleast once regardless the condition is true or not.

25/07/2022

(01.) By using do-while loop write a program to allow the user to input 10 numbers and display the total count of odd, even numbers in the entered number series.

```
int counter = 1, oddc = 0, evenc = 0, no;
do
{
    printf("Enter %d number ", counter);
    scanf("%d", &no);
    if (no%2 == 0)
        evenc++   //evenc = evenc + 1
    else
        oddc++   //oddc = oddc + 1
    counter++;
}while (counter <= 10);

printf("The total odd count is %d \n", oddc);
printf("The total even count is %d \n", evenc);
```

Date No

(02) Write a program to allow the user to input two numbers and display the output of basic arithmetic operations. in the program, user can repeatedly select the arithmetic calculation.

```
//allow the user to input two numbers.
printf ("Enter two numbers ");
scanf ("%d %d", &no1, &no2);

do{
    //display the menu
    printf ("1.+ \n 2.- \n 3.* \n  4./ \n 5. Exit \n");
    printf("Select an option");
    scanf("%d", &opt);

    switch(opt)
    {
     case 1 : printf ("The addition is %d \n", (no1+no2)); break;
     case 2 : printf("The subtraction is %d \n", (no1-no2)); break;
     case 3 : printf("The multiplication is %d \n", (no1*no2)); break;
     case 4 : printf("The division is %d \n", (no1/no2)); break;
    }
}while (opt!=5);
```

Date No

For Loop

(01.) 1 2 3 4 ... 100

```
int x;
for(x=1; x<=100; x++)      *no semicolon at the end
    printf("%d", x);
```

(02.) 100 98 96 ... 2

```
int x;
for (x=100; x>=2; x=x-2)
    printf("%d", x);
```

(03) 5 10 15 ... 100

```
int x;
for(x=5; x<=100; x=x+5)
    printf("%d", x);
```

Nested For Loops.

(01.) By using nested for loops display the following output.

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
int x, y;
for (x=1; x<=5; x++)
{
   for (y=1; y<=5; y++)
   {
      printf ("* ");
   }
   printf ("\n");
}
```

(02.) By using nested for loops display the following output.

column ↓

row →

```
*
*   *
*   *   *
*   *   *   *
*   *   *   *   *
```

```
//nested For loops
int r, c;
for (r=1; r<=5; r++)
{
   for (c=1; c<=r; c++)
   {
      printf("* ");
   }
   printf ("\n");
}
```

Date 01/08/2022

Arrays.

* An array is a data structure.
* We can use arrays to store set of values with similar data types.
* In C language we discuss the following 2 types of arrays
  01. Single - Dimensional Arrays. (Vectors)
  02. Multi - Dimensional Arrays. (Matrix)

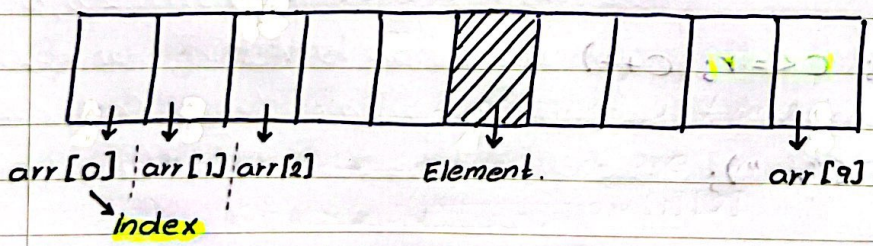
Single - Dimensional Arrays.

(01.) Declare a single - dimensional array to store 10 integer values. Input 10 values into the array and display the values.

I. How to declare an array?

```
int   arr[10];
```

Data type — Array name — Array size



* In an array the elements are uniquely identified by using an index.

Date No

* The index always begins with 0 and ends with (array size - 1)

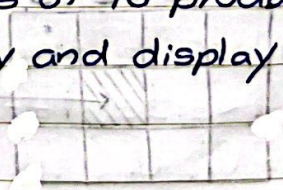
II. You can input values into an array by using a loop.

III. Again you can use a loop to display values.

```
{
//declare
int arr[10];
//input
int i;
for(i=0; i<10; i++)
{
    printf("Enter a number ");
    scanf("%d", &arr[i]);
}
//print
printf("The array values are \n");
for(i=0; i<10; i++)
    printf("%d ", arr[i]);
}
```

12/08/2022

01) Declare an array to store prices of 10 products. (float) Input the prices into the array and display the highest price and the average price.

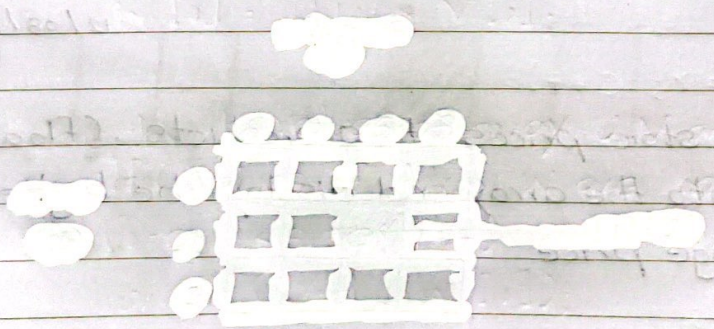


Date No

```
{
    float prices [10], sum = 0, max = 0, avg;
    int i;

    For (i=0; i<10; i++)
    {
        printf ("Enter a value to the element %d ", i+1);
        scanf ("%f", &prices [i]);
        sum = sum + prices [i];
        if (prices [i] > max)
            max = prices [i];
    }
avg = sum / 10.0;
printf ("The highest price is %.2f \n", max);
printf ("The average price is %.2f \n", avg);
}
```

Multi-Dimensional Arrays.

(01) Declare a multi-dimensional array with 3 rows and 4 columns. Input values in to the array and display the values, in the form of matrix.



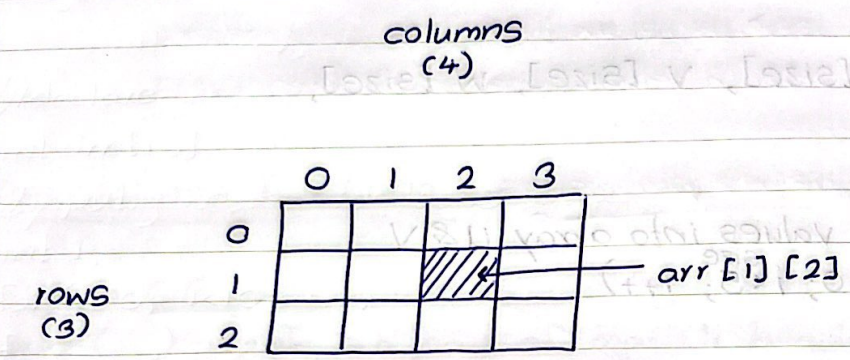
Date No

Syntax;

<Data Type> <name> [<no. of rows>] [<no. of columns>]

Ex;

int arr [3] [4]




```
{
  int U [3] [4], ri, ci;  // ri = row index, ci = column index
  for (ri=0; ri<3; ri++)
  {
     for (ci=0; ci<4; ci++)
     {
        printf ("Enter a Value" );
        scanf ("%d", &U [ri] [ci]);
     }
  }
  For (ri=0; ri<3; ri++)
  {
     for (ci=0; ci<4; ci++)
     {
        printf ("%d", U [ri] [ci]);
     }
     print f ("\n");
  }
}
```

Date No

(02.) Declare two single dimensional arrays with the size of 5. Input values into the array and display the products of related elements in the third array. Display the values stored in all 3 arrays.

```
#include <stdio.h>
#define size 5    //because size is a constant.
int main ()
{
   int u [size], v [size], w [size];
   int i;

   //input values into array u & v
   for (i=0; i<size; i++)
   {
      printf ("Enter a value to element %d of Array u", i+1);
      scanf ("%d", &u [i]);
      printf ("Enter a value to element %d of Array v", i+1);
      scanf ("%d", &v [i]);
   }

   //store the product in the array w
   for (i=0; i<size; i++)
       w [i] = u [i] * v [i];

   //display the values of array u, v and w
   printf ("Values of Array w are \n");
   for (i=0; i<size; i++)
       printf ("%d", w [i]);
}
```

Date 15/08/2022

Functions.

* A function in C language has its on task to perform.
  Ex: printf( ); and scanf( ); which is use to display and input data.

* The functions are broadly divided into 2 catergories.
  1) Pre-defined functions.
  2) User-defined functions.

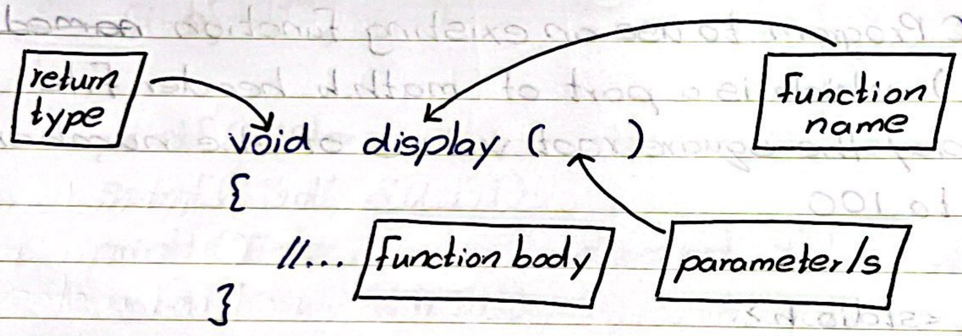
Pre-defined functions.

(01) Write a C Program to use an existing function sqrt( ) which is a part of math.h header file and display the square root values of the numbers From 1 to 100

```
#include <stdio.h>
#include <math.h>
int main ()
{
    float x, ans;

    for (x=1; x<=100; x++)
    {
        ans = sqrt (x); // here sqrt () is the pre-defined function.
        printf ("Square Root Value of %.2f is %.2f \n", x, ans);
    }
}
```

User-defined functions.

* In the user-defined function, we have to provide the logic of the program.
* There're 4 catergories of user-defined functions.
  1) no return type, no parameters.
  2) no return type, with parameters.
  3) with return type, no parameters.
  4) with return type, with parameters.



*void තිබ්බොත් no return type.
නැත්නම් there's a return type
Ex: int, float.

a) No return type, no parameters.

```
#include <stdio.h>

void display ()
// A function with no return type and no parameters.
// Called function.

{
    int x;
    for (x=1; x<=10; x++)
        printf ("C is easy to learn \n");
}
```

```
int main ()
{
    //call the above function
    display ();
    display();
}
```

* In the above program we're calling the display function, inside the main method.
* You can call the function display inside the main method, any number of times.
* The main advantage of function is, its reusability.

(01.) Write a C program to create the following two functions.
01.) A function to display your name 20 times.
02.) A function to display your school name 10 times.
After creating the functions call the above functions inside the main method.

```
#include <stdio.h>

void displayName()
{
    int x;
    for(x=1; x<=20; x++)
        printf("Afsaan \n");
}
```

Date No

```
void displaySchool ()
{
    int x;
    for (x=1; x<=10; x++)
        printf ("NSBM Green University \n");
}

int main ()
{
    displayName ();
    displaySchool ();
}
```

b) No return type, with parameters.

(01) Create a function which accept two integers as parameters and display the total.

```
#include <stdio.h>

void findSum (int a, int b)
{
    int sum = a+b;

    printf ("The total is %d \n", sum);
}

int main ()
```

Date No

```
{
    int x, y;

    printf("Enter two numbers ");
    scanf("%d %d", &x, &y);

    findSum(x, y);
}
```

(02.) Write a program to create a function to allow the user to input 2 numbers as parameters and display the highest number.

```
#include <stdio.h>

void findHighest(int a, int b)
{

    if (a > b)
        printf("%d is the highest \n", a);
    else
        printf("%d is the highest \n", b);
}

int main()
{
    int x, y;

    printf("Enter two numbers ");
    scanf("%d %d", &x, &y);
    findHighest(x, y);
}
```

ProMate

Date No

(03) Create a function to allow the user to pass an integer parameter and display the entered number is a positive, negative or zero.

```
#include <stdio.h>

void check (int a)
{
    if (a > 0)
        printf ("%d is a positive number \n", a);
    else if (a < 0)
        printf ("%d is a negative number \n", a);
    else
        printf ("%d is a zero \n", a);
}

int main ()
{
    int x;

    printf ("Enter a number ");
    scanf ("%d", &x);

    check (x);
}
```

Date 22/08/2022

01.) Create a function which accept marks as a parameter, find and display the grade of the module. The grade is calculated as per the following table.

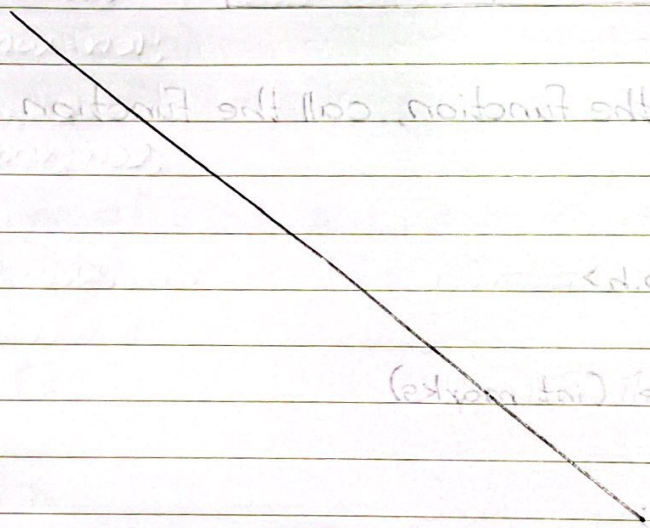
Marks	Grade
⩾75	A
50-75	B
<50	F

After creating the function, call the function inside the main method.

```
#include <stdio.h>

void findGrade (int marks)
{
    char grade;

    if (marks >= 75)
        grade = 'A';
    else if (marks >= 50)
        grade = 'B';
    else
        grade = 'F';

    printf("The grade is %c \n", grade);
}
```

Date No

```
int main ()
{
    int m;

    printf("Enter your module mark ");
    scanf("%d", &m);

    findGrade (m);
}
```



22/08/2022

9) With return type, no parameters.

* In a function, return type means any data type in C language other than the term "void".

(01.) Create a function which allows the user to input two integers and return the total.

* [redacted] return type එකක් නිසා ඔබෙන්ම last line එක return.

ProMate

```c
#include <stdio.h>

int findSum ()
{
    int no1, no2, sum;

    printf ("Enter two numbers ");
    scanf ("%d %d", &no1, &no2 );

    sum = no1 + no2;

    return sum;
}

int main ()
{
    printf ("The sum is %d \n", findSum ());
}
```

* A function with return type always returns a single value.
* All the functions with return type should be called inside the printf.

(02) Create a function which allow the user to input two numbers and return the highest number.

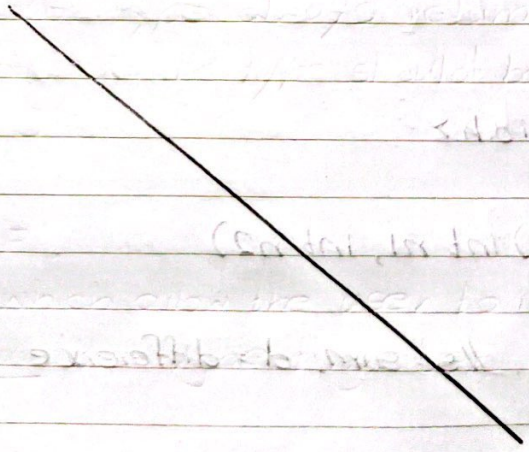
Date No

```
#include <stdio.h>

int findMax()
{
    int no1, no2, max;

    printf("Enter two numbers ");
    scanf("%d %d", &no1, &no2);

    if (no1>no2)
        max=no1;
    else
        max=no2;

    return max;
}

int main()
{
  printf("The highest is %d \n", findMax());
}
```

(01.) Create a function which accept an integer as a parameter and return the number as an odd or even number.

```
#include <stdio.h>

int findOE (int n)
{
    if (n%2==0)
        return 0;
```

```
    else
        return 1;
}

int main ()
{
    int n, ans;

    printf ("Enter a number ");
    scanf ("%d", &n);

    //here we call the function findOE() and it returns a
                                                    number.

    ans = findOE (n);

    if (ans == 0)
        printf ("This is an even number ");
    else
        printf("This is an odd number ");
}
```



Date No

Tutorial 07

(01.)
```
#include<stdio.h>

void display ()
{
    int n1, n2, s, d;    //s = sum, d = difference

    printf("Enter two numbers ");
    scanf("%d %d", &n1, &n2);

    s = n1+n2;
    d = n1-n2;

    printf("The sum is %d \n", s);
    printf("The difference is %d \n", d);
}

int main ()
{
    display ();
}
```

(02.)
```
#include <stdio.h>

void display ( int n1, int n2)
{
    int s, d;  //s = sum, d = difference

    s = n1+n2;
    d = n1-n2;
```

ProMate

```
    printf("The sum is %d \n", s  );
    printf("The difference is %d \n", d);
}

int main ()
{
    int a, b;

    printf("Enter two numbers ");
    scanf("%d %d", &a, &b);

    display (a,b);
}
```

(03)

```
#include <stdio.h>

int findPro (int a, int b)
{
    int p;

    p = a * b;

return p;
}

int main ()
{
    int n1, n2;

    printf("Enter two numbers ");
    scanf("%d %d", &n1, &n2);

    printf("The Product is %d \n", findPro (n1, n2));
```

*Product = ලබාගැනීම

(04.)
```
#include <stdio.h>

float findQuo (int a, int b)
{
    float q;

    q = (float)a / (float)b;

    return q;
}

int main ()
{
    int n1, n2;
    float quot;

    printf("Enter two numbers ");
    scanf("%d %d", &n1, &n2);

    quot = (float)n1 / (float)n2;

    printf("The Quotient is %.2f \n", findQuo(n1,n2));
```

* Quotient = ලබ්ධ.

(05.)
```
#include <stdio.h>

void display ()
{
    int n1, n2, sum;

    printf("Enter two numbers ");
    scanf("%d %d", &n1, &n2);
```

```
    sum = n1+n2;

    printf("The sum is %d \n", sum);
}

int main ()
{
    display ();
    display ();
    display();
}
```

(06)

```
#include <stdio.h>

void display (int n1, int n2)
{
    int s, d, p;  //s = sum, d = difference, p = product

    s = n1 + n2;
    d = n1 - n2;
    p = n1 * n2;

    printf("The sum is %d \nThe difference is %d \n The
                product is %d \n", s, d, p);
}

int main ()
{
    int a, b;
```

```
    printf ("Enter two numbers ");
    scanf ("%d %d", &a, &b);

    display (a, b);
}
```

(07)
```
#inolude <stdio.h>

double findPro (int a, float b)
{
    double p;

    p = (double) a * (double) b;

    return p;
}

int main ()
{
    int n1;
    float n2;

    printf ("Enter two numbers ");
    scanf ("%d    %f", &n1, &n2);

    printf ("The Product is %f \n", findPro (n1, n2));
}
```

Date 24/08/2022

Recursion.

* Recursion is a special type of function.
* In recursion the function calls by itself.
* This is closely associated with the iteration. (Repitition)
* The iteration is an explicit loop where as recursion is a implicit loop.

```
#include <stdio.h>

void display()
{
    printf("Hello World!");
    //function calls by itself (recursion)
    display();
}

int main()
{
    display();
}
```

(01.) Write a program using loops to find a factorial value of a give number.
For the above create a function which accept a number as a parameter and return the factorial value.

* Factorial Value

Ex: 5

$5! = 5 \times 4 \times 3 \times 2 \times 1$

$= 120$

```
#include <stdio.h>

int findFact (int n)
{
    int ans = 1;
    for (x = 1; x <= n; x++)
        ans = ans * x;

    return ans;
}

int main ()
{
    printf ("The answer is %d \n", findFact (5));
}
```

* The above program is written using loops. But the same program can be implemented by using Recursion without loops.

Pointer

* A pointer is a special type of variable.
* It is used to display the memory address of an existing variable.

Date No

```c
#include <stdio.h>

int main ()
{
    //a variable
    int x=10;
    //a pointer
    int *p;
    //this is to store the memory address of an existing variable.
    p = &x;
    //display the memory address
    printf("Memory address of x is %p \n", p);
}
```