SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF DATASCIENCE AND BUSINESS SYSTEMS

18CSC161J - Fundamentals of Computer Science

STUDENT PORTFOLIO

Insert Photo

Name: DHAWAL SAHU

Register Number: RA2111042010015

Mail ID: dc5748@srmist.edu.in

Department: Computer science

Specialization: Business System

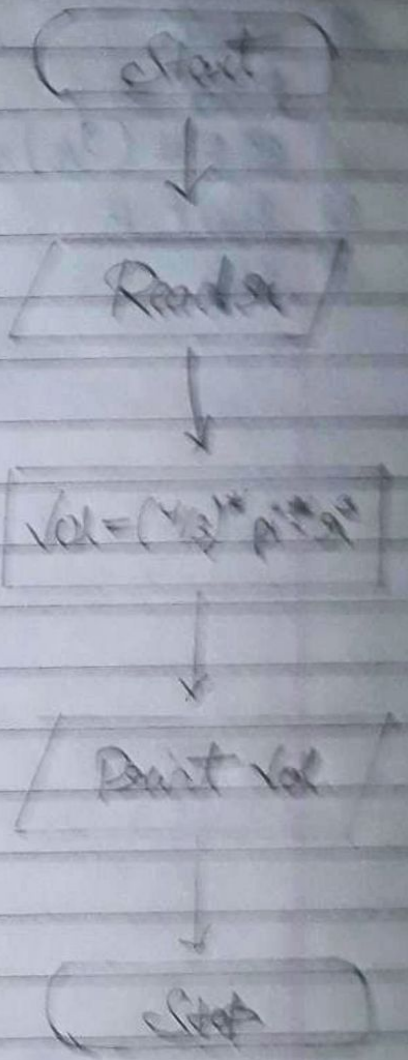
Semester: Ist

Faculty In-Charge : Dr K. SHANTHA KUMARI

1 Algorithum

i) Start
ii) Read r
iii) Vol = (4/3) * pi * r^3
iv) Print vol
v) Stop

Flowchart

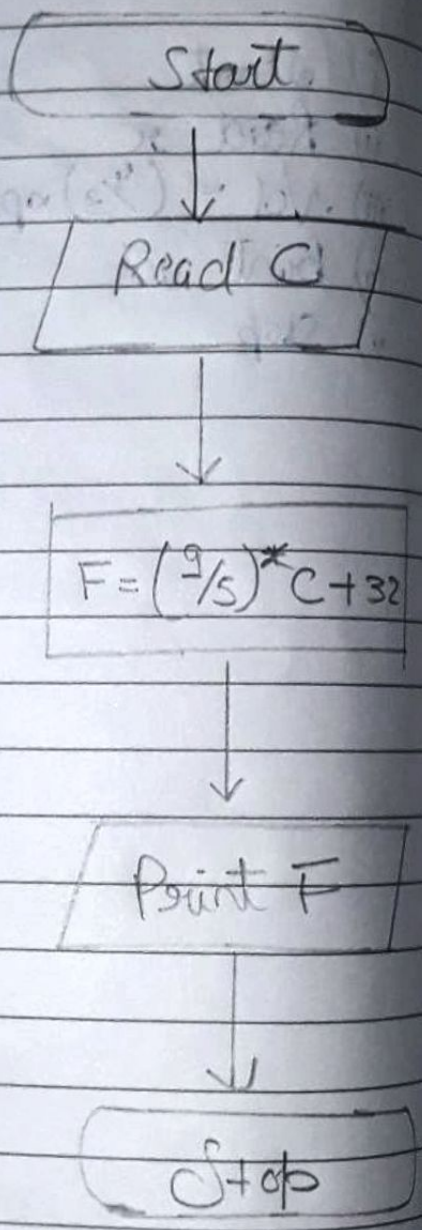


Date / /

2 Algorithm

i) Start
ii) Read C
iii) F = (9/5) * C + 32
iv) Print F
v) Stop

flow chart

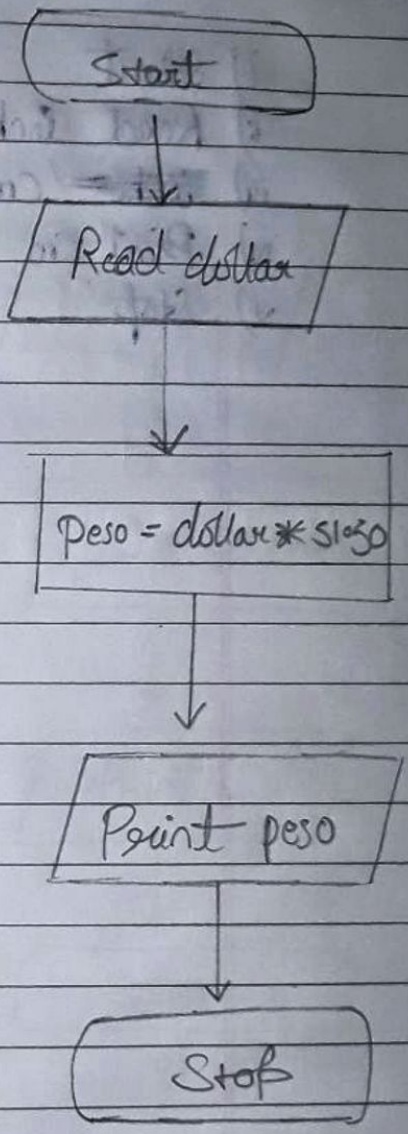


Page No.
Date

3 Algorithm

i) Start
ii) Read dollar
iii) Peso = dollar * 51.50
iv) Print peso
v) Stop

flow chart

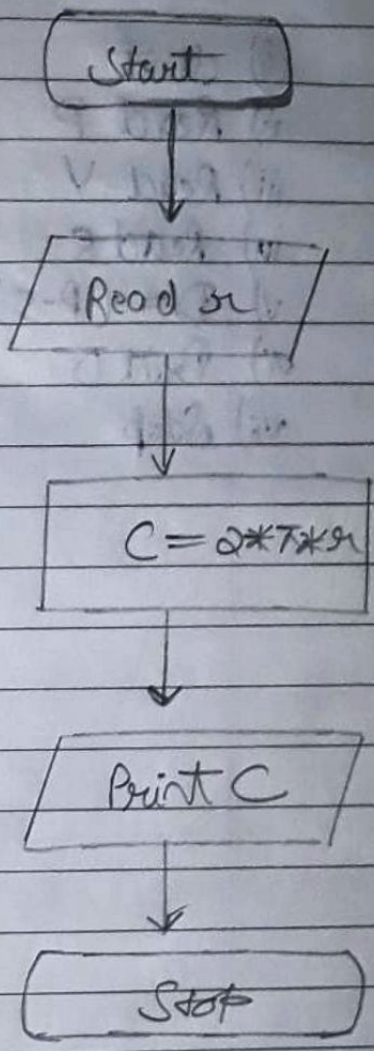


Page No.
Date / /

5 Algorithm

i) Start
ii) Read r
iii) $C = 2\pi r$
iv) Print C
v) Stop

flowchart

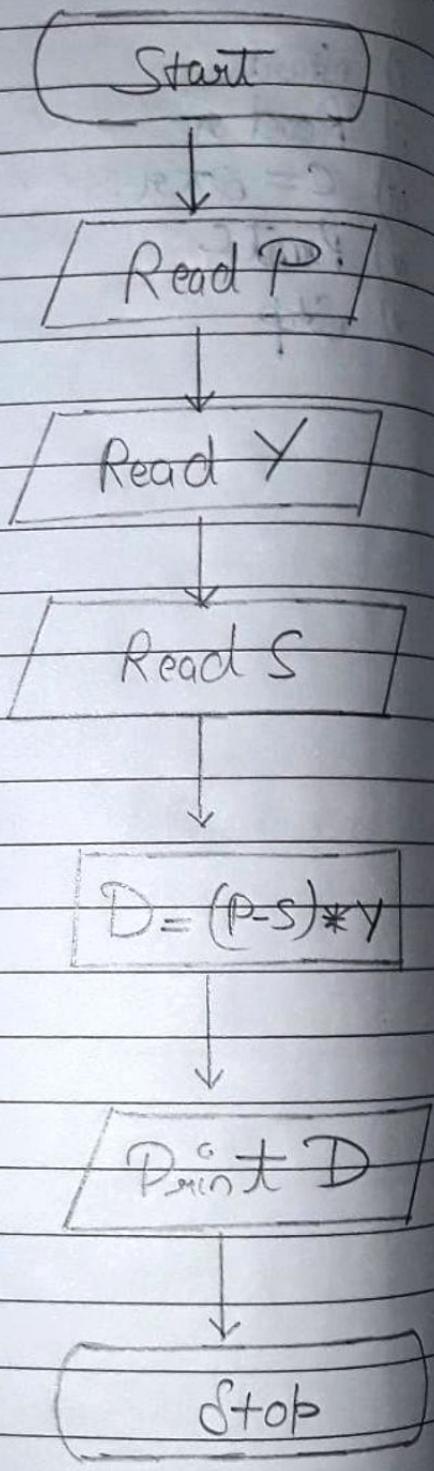


Date / /

6 Algorithm

i) Start
ii) Read P
iii) Read Y
iv) Read S
v) D = (P-S)*Y
vi) Print D
vii) Stop

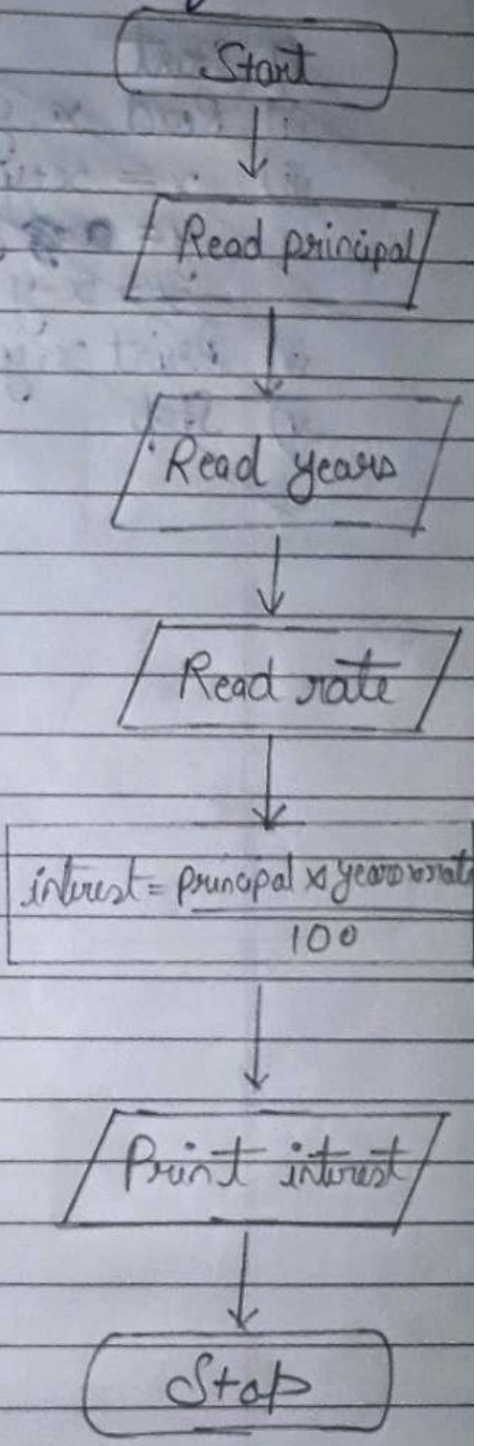
flowchart



7 Algorithm

i) Start
ii) Read principal
iii) Read years
iv) Read rate
v) Interest = Principal * years * rate /100
vi) Print interest
vii) Stop

flowchart

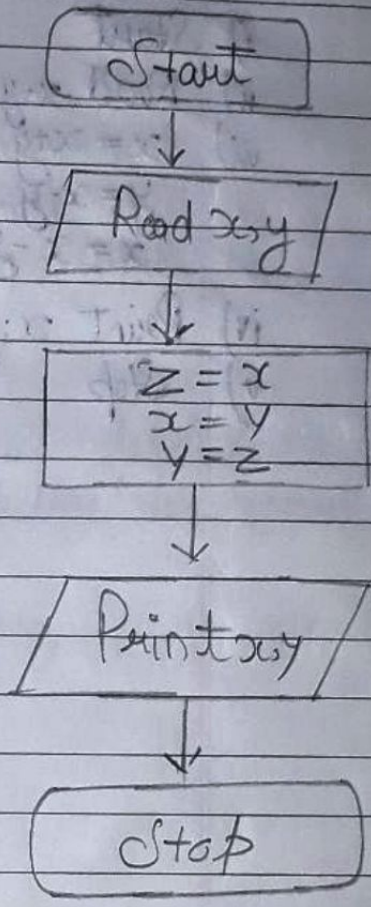


Page No. /
Date / /

8. Algorithm

i) Start
ii) Read x, y
iii) Declare third variable z
$z = x$
$x = y$
$y = z$
iv) Print x, y
v) Stop.

flow chart



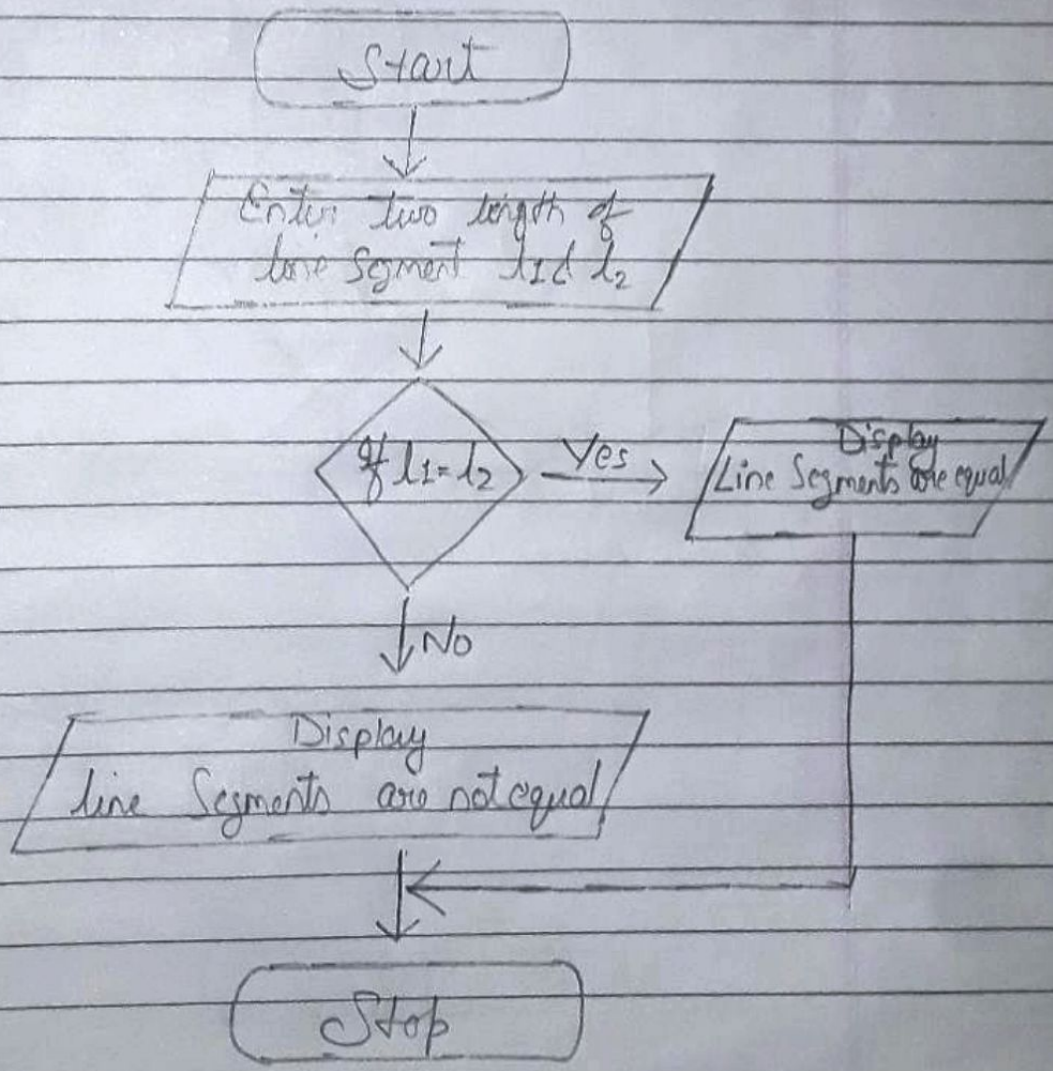
10 Algorithm

flowchart

i) Start
ii) Accept the line segment

10 Algorithm

i) Start
ii) Accept the length of two line segment l_1 & l_2
iii) If l_1 & l_2 are equal, then display 'line segment are equal'
iv) If l_1 & l_2 are not equal then display, line segment are not equal'.
v) stop.

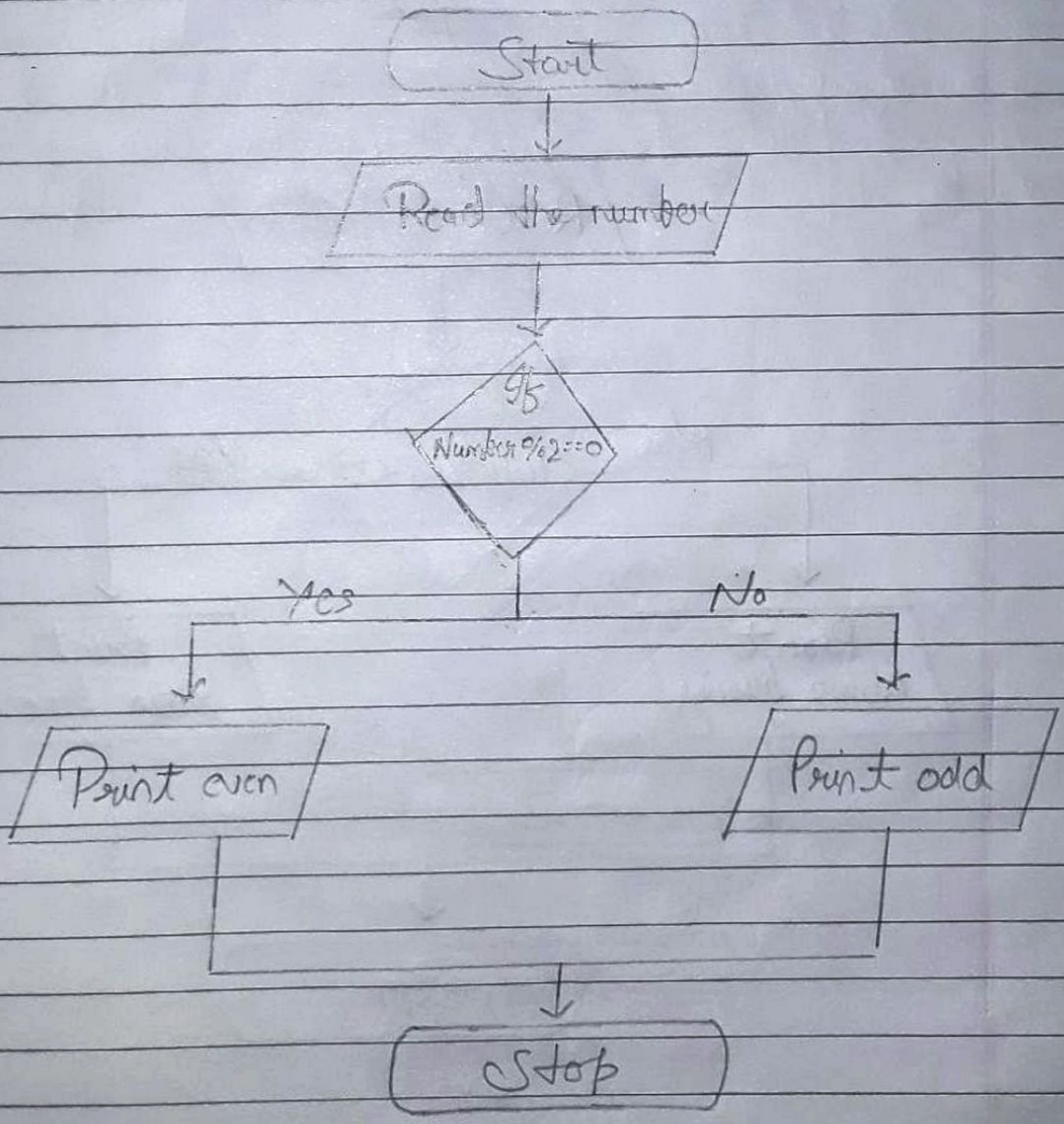


Page No. /
Date / /

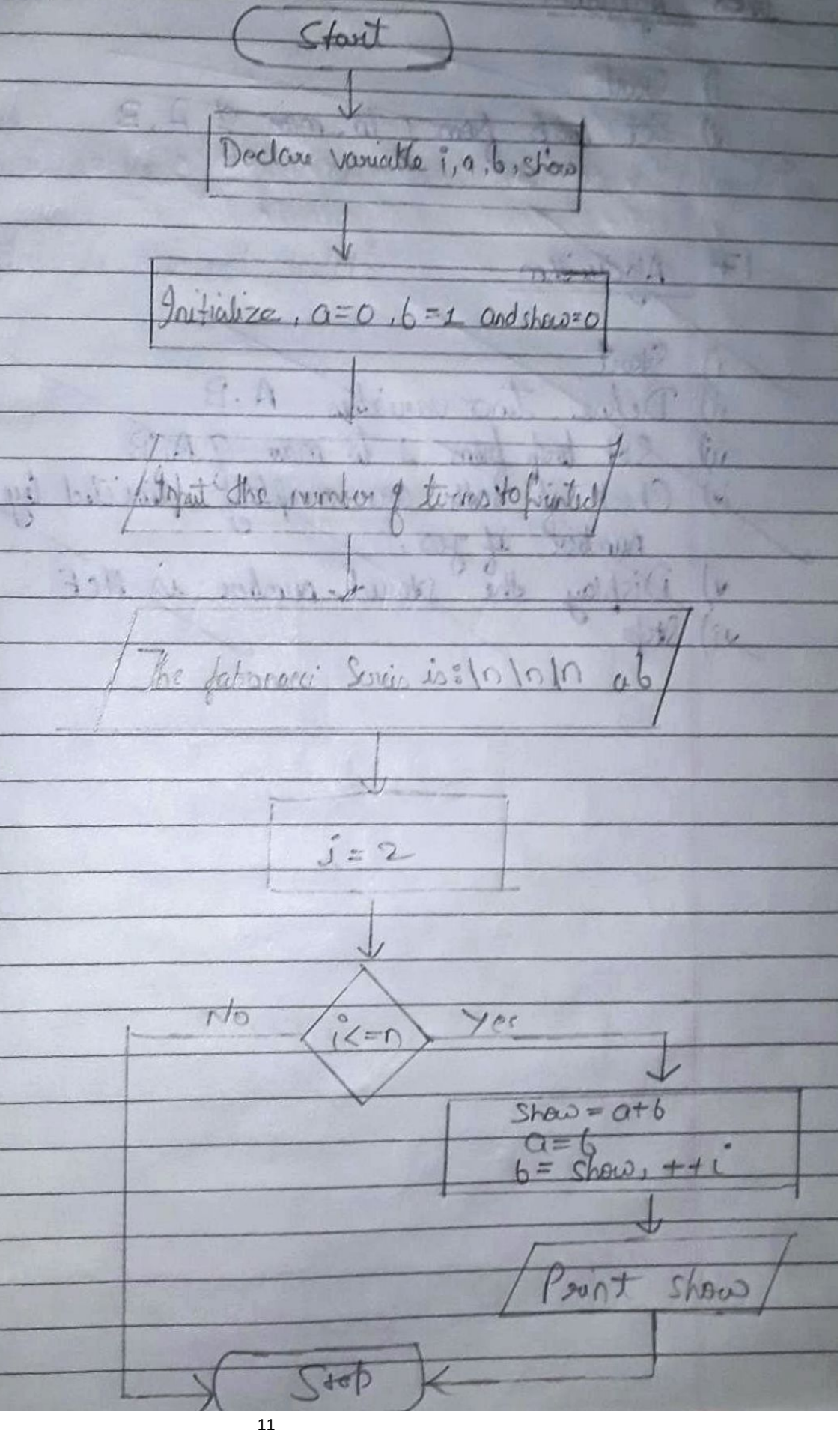
1. Algorithm

i) Start
ii) Read the number
iii) If number %2==0 then number is even
iv) Else number is odd
v) print output
vi) Stop

flowchart



14 flow chart

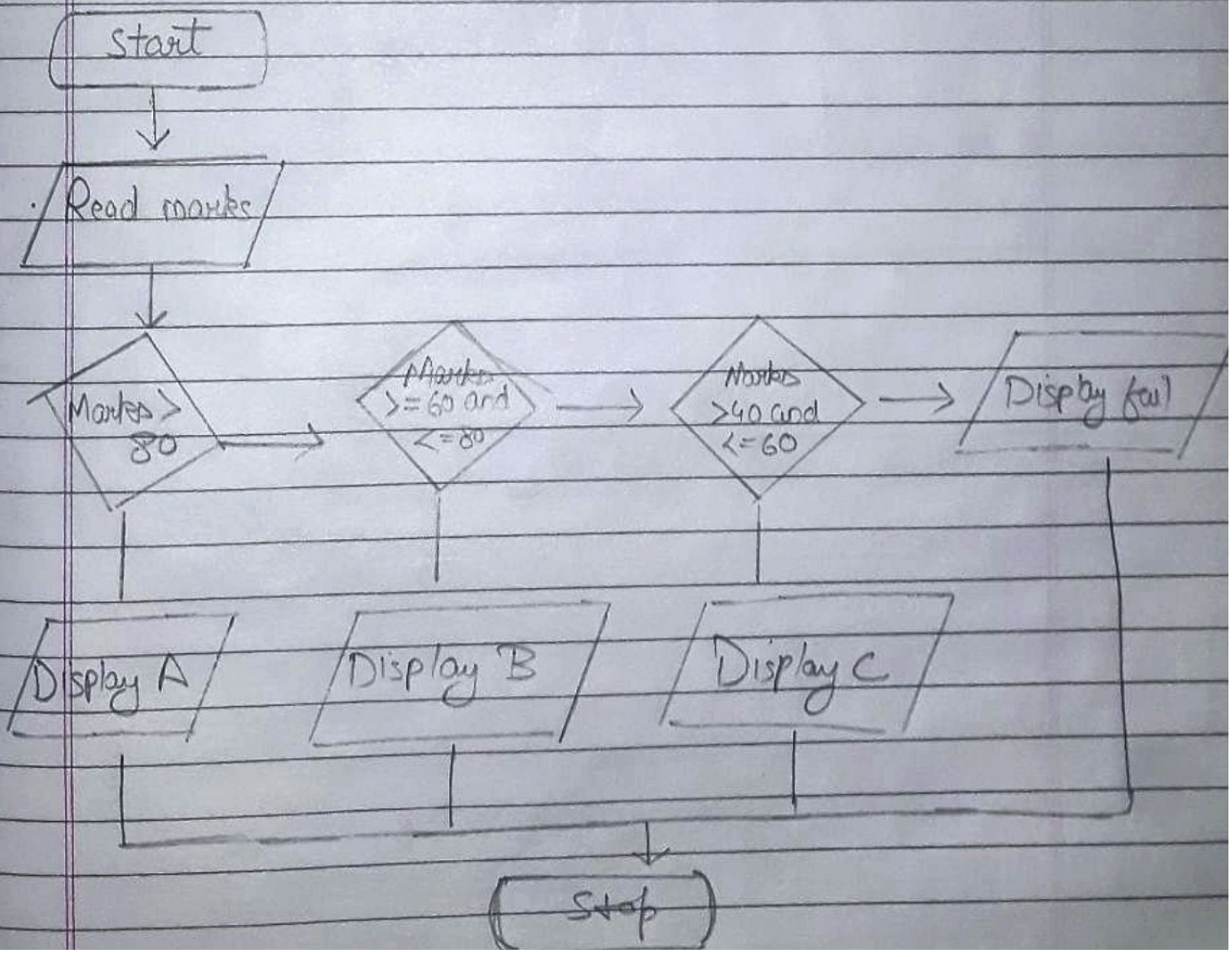


Page No. /
Date / /

2.1 Algorithm

i) Start
ii) Read Marks
iii) if Marks >= 80 then grade = A, go to step 7
iv) if marks >= 60 and marks <= 80 then grade = B, go to step 7
v) if marks >= 40 and marks <= 60 then grade = C go to step
vi) print No Grade
vii) print grade
viii) Stop.

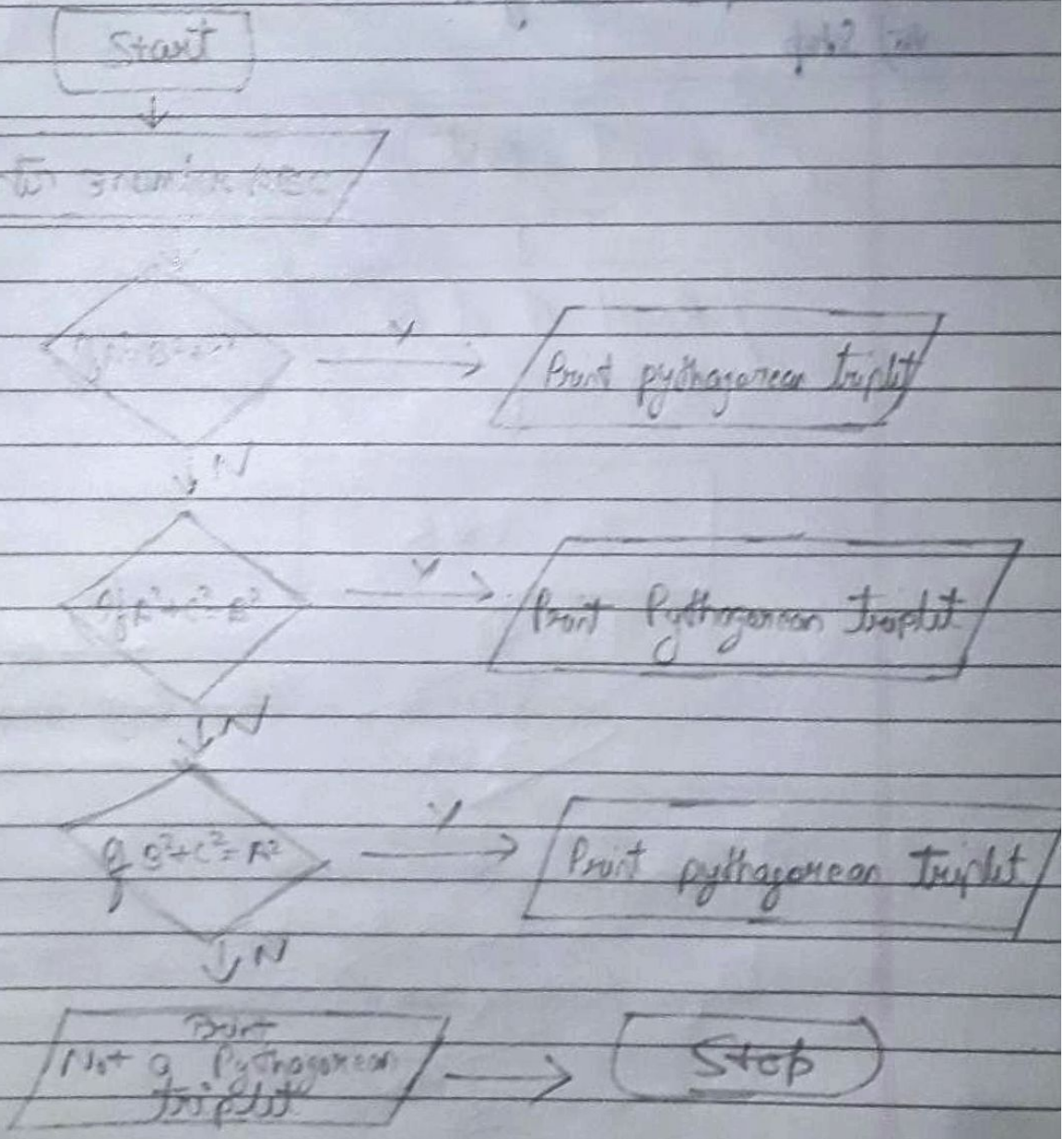
flow chart



Page No. /
Date / /

20 Algorithm

i) Start
ii) Take 3 numbers as input A, B, C
iii) If $A^2 = B^2 + C^2$ then print "Pythagorean triplet"
iii) If $B^2 = A^2 + C^2$, then print "pythagorean triplet"
v) If $C^2 = A^2 + B^2$ then print "pythagorean triplet"
vi) If no condition of step 3, 4, 5 the print Not pythagorean triplet.
vii) stop



Page No. /
Date / /

Programming :-

i) Programming has its origin in the 19th Century, when the first 'programmable' looms and Player piano scrolls were developed.

ii) In 1940's the modern computers & have been introduced and programming languages were developed.

iii) The first high level programming language to be designed for a computer was Plankalkül, developed for the German Z3 by Konrad Zeuse between 1943 & 1945.

Programming language Category:

i) Machine language
ii) Assembly language
iii) High level language

Steps to solve problem using programming:-

i) Understand the program correctly
ii) Analyze, divide & assess
iii) Optimize the steps
iv) Write Pseudo Code
v) Write program
vi) Optimize Code

Page No. /
Date / /

Algorithm:- A finite sequence of well defined instructions to solve a problem with the help of computer is called as algorithm.

Characteristics :-

Unambiguous:- Each step should be defined in precise manner.

Finite: The algorithm should terminate after a finite no. of steps and each steps should take a finite time.

Unique:- Each step should be unique defined and only depend on the input and the result of the preceding steps.

Q Sum of two Numbers.

Step I Start
Step II Declare variable num1, num2 & sum
Step III Read values num1 & num2.
Step IV Add num1 & num2 and assign the result to sum
(sum ← num1 + num2)
Step V Display sum
Step VI Stop

Page No. /
Date / /

Guidelines to create a flowchart :-

i) It should be clear, neat and easy to follow
ii) It must have a logical start & finish.
iii) In drawing a proper flowchart all necessary requirement should be listed in logical order.
iv) The direction of the flow of procedure should always be from left to right or top to bottom.
v) Only 1 flowline should come out from a process symbol
vi) Only one flowline should enter a decision symbol. However 2 or 3 flowlines may leave the decision symbol.
vii) Only one flowline is used with a terminal symbol
viii) Within standard symbol annotation symbol are used to describe the process.
ix) In case of complex flowchart connectors symbols are used to reduce the number of flowlines.
x) Intersection of flowlines should be avoided.
xi) It is useful to test the validity of the flowchart by passing through it with normal/unusual test data.

Benifits of flowchart :-

i) Pictorial representation
ii) Coding can be done faster
iii) Helps detecting errors
iv) Good program documentation tool
v) Redundant steps identification
vi) Misplaced steps identification

Page No. /
Date / /

Limitations

i) Complex
ii) Costly
iii) Difficult to Modify
iv) No update

Pseudocode :- It does not follow the syntax of any specific programming language but has some features that would be similar to programming language

Some keywords -

i) Input : READ, OBTAIN, GET, PROMPT
ii) Output : PRINT, DISPLAY, SHOW
iii) Compute : COMPUTE, CALCULATE, DETERMINE
iv) Initialise : SET and INITIALISE
v) Add one : INCREMENT.

Guidelines :-

i) Programming language independent
ii) Logic plan to develop a program
iii) Should produce the solution to the specified pro
iv) Easy for a person to read & translate into cod
v) Concise and each instruction should be written a seperate line

Page No. /
Date / /

Benefits:

i) Express the design in plain natural language
ii) Programming language independent
iii) Easier to develop a program
iv) Compact

Limitations

i) Do not provide visual representation of the program's logic
ii) No accepted standard for writing pseudo code
iii) Cannot be compiled nor executed.

INPUT & OUTPUT FUNCTION:-

Formatted INPUT:-

i) Scanf() is a short form of scan formatted and used to read all types of data values. It is used for runtime assignment of values for variables.

Formatted OUTPUT:-

i) printf() is the short form formatted and is used to print all types of data values on to the standard output device.

Page No
Date / /

VARIABLES:-

i) Naming -

→ A variable can be any combination of alphabets, digits and underscore. Length of the variable can be from 1 to 31 character, though some compiler allow bigger names, it is better to stick to not going more than 31 characters.

→ Variables should not start with a digit.

→ The first character must be an alphabet or an underscore (_). However is it better to avoid variables starting with _, as they are generally used in library routines.

→ No commas or blank spaces are allowed.

→ No special symbols other than underscore can be used in variables.

→ C is case sensitive and hence capital letter is considered to be different from small letter.

→ Key words cannot be used as variables names.

Data types and sizes

Type	Size	Range	Format specifier
Char	1	-128 to 127	%c
int	2	-32768 to 32767	%d or
float	4	3.4e-38 to 3.4e+38	%f or
double	8	1.7e-308 to 1.7e+308	%lf

Page No. /
Date / /

* **Declaration** :- **Syntax** - <data_type> list of variable name seperated by comma;

Ex- int age;

* **Place of declaration** :-

→ Inside a function or a block which is called local variables
→ Outside of all functions which is called global variables.
→ In the defination of function parameters which are called formal parameters.

Static initialization :-

Variable is assigned a value and it acts as a Const. That is every time the program is run, it initialize to the same value.

Ex - int age = 20, roll no = 75;

Dynamic initialization :-

Variable is assigned a value at run time.

```
Ex - int age
     printf("Enter your age.\n");
     Scanf("%d", age);
```

Constants :- Entity whose value doesn't change during the execution of the program.

Type Conversion :-

→ Automatic Conversion happens when an expression has more than one data type to be present. All the data types are upgraded by Compiler to that of the largest data type. It is possible to lose information when automatic conversion happens.

→ Explicit Conversion can be done using typecast
Syntax: (type) expression

Modifiers :-

→ Long : Used to increase size of the current data type to 2 more bytes.
→ short : Used to allocate fixed memory space using of the underlying operating system.
→ Unsigned → used to make the accepting value of data type is positive data type.
→ Signed → Used to accept both negative or positive value and this is default.

Page No /
Date / /

Standard header files :-

Stdio.h → Standard input output function
Conio.h → Console input / out function
Stdlib.h → General utility function
math.h → Mathematical function
String.h → string function
Ctype.h → Character handling function
time.h → Date and time function

Structure of program :-

i) Documentation
ii) Header
iii) Global declaration
iv) main() function {
v) Local declaration
vi) executable statement }
vii) User difined function

Operators in C :-

Unary operator :-

i) Unary plus +
ii) Unary minus −
iii) Increment operator ++
iv) decrement operator --
v) Address of operator &
vi) Size of operator size of ()
vii) Dereferencing operator *
viii) Logical Not !

Page No. /
Date / /

⑤ Relational operator
→ Binary operator
→ Produces an integer result
* Condition true: Integer value is 1
* Condition false: Integer value is 0
→ Compares
* Value b/w two variables
* Values b/w variables and constants

⑥ Logical operator
→ Combines two or more relations
→ Used for testing one or more conditions

Symbols
"&&" Logical AND – true when all expression are T
"||" Logical OR – true either expression is T
"!" Logical NOT – negation

op(1)	Op(2)	Op(1) && Op(2)	Op(1) || Op(2)
F (0)	F (0)	F (0)	F (0)
F (0)	T (1)	F (0)	T (1)
T (1)	F (0)	F (0)	T (1)
T (1)	T (1)	T (1)	T (1)

Page No.
Date

(C) Conditional operator
⇒ ? and : are the Conditional operator
→ Also called as Ternary operators
→ Shorter form of if-then-else statement

Bitwise operator

i) & Bitwise AND
ii) | Bitwise OR
iii) ^ Bitwise XOR
iv) ~ Bitwise Compliment
v) << Shift left
vi) >> Shift right

Operator precedence is used to determine the order operator evaluated in an expression.

Associativity is used when two operators of same precedence appear in an expression.

Expressions using pre/post increment operators:-
→ Increment operator increases the value of the variable one
→ Decreases the operator decreases the value of the variable by one.

Page No.
Date

Expressions using assignment operator:-

→ Assignment operator is used to assign value to an variable
→ Assignment operator denoted by equal to sign.
→ Assignment operator is binary operator which operates on two operands.

Control statement

→ Also called as conditional statement
→ Decides order of execution based on condition
→ Helps repeat a group of statement
→ Modifies control flow of program
→ Decision making
→ Branching

Types of Branching statement :-

a) If statement
b) if else statement
c) Nested if else statement
d) else if statement

→ Switch statement
→ goto statement

Page No.
Date

Unit III

Function → A function is a self contained block of statements that perform a coherent task of some kind. Every C program can be thought of as a collection of these functions.

Suppose we have a task that is always performed in the same way - Eg birthday servicing of your motorbike. You don't need to give instructions to the mechanic because he knows his job.

Let us look at simple C function that operates in much the same as the mechanic. We will be looking at two things - a function that calls or activate the function and the function itself.

```
#include <stdio.h>
void message();
int main()
{
  message();
  printf("Cry, and you stop the monotony\n");
  return 0;
}
void message()
{
  printf("Smile, and the world smiles with you...\n");
}
```

Output -

Smile, and the world smiles with you
Cry, and you stop the monotony!

Page No. /
Date / /

* A C program is a collection of one or more function
* If a C program contains only one function, it must be main()
* If a C program contains more than one function, then one of these function must be main(), because program execution always being with main().
* There is No limit on the number of function that might be present in a C program.
* Each function in a program is called in the sequence specified by the function calls in main().
* After each function has done its things, control return to main(), when main() runs out of statement and function calls, the program ends.

(a) A function gets called when the function name is followed by a semicolon.
(b) A function is defined when function name is followed by a pair of braces ({}) in which one or more statement may be present.
(c) Any function can be called from any other function. Even main() can be called from other function.
(d) A function can be called any number of times
(e) The order in which function they get called need not necessary be same.
(f) After a function can call itself. Such a process is called 'recursion'. We would discuss the aspect of C
(g) A function can be called from another function, but a function cannot be defined in another function. Thus, the following a program code would be wrong,
(h) There are 2 types of function
Library function → printf(), scanf()
User defined function → argentina(), brazil().

Page No.
Date / /

Passing values between function

The mechanism used to convey information to the function is the 'argument'. You have undoubtedly used the argument in the printf() and scanf() function. The format string and the list of variables used inside the parentheses in these functions are arguments.

Using Library function :-

```
#include <stdio.h>
#include <math.h>
int main()
{
  float a=0.5;
  float w,x,y,z;
    w = sin(a);
    x = cos(a);
    y = tan(a);
    z = pow(a,2);
    printf("%f %f
```

Here we have called four standard library functions sin(), cos(), tan(), and pow().
Before calling any function, we must declare its prototype. Since we didn't define the library functions, we don't know the prototype declaration of library function. When the library function is provided, a set of .h files also provided.

Page No. /
Date / /

Prototype of functions sin(), cos(), tan(), pow() are declared in the file 'math.h'

* Pointers are variables which hold addresses of other variables
* A pointer to a pointer is a variables that holds address of a pointer variable.
* The & operator fetches the address of the variable in memory.
* The * operator lets us access the value present at an address in memory with an intension of reading it or modifying it.
* A function can be called either by value or by reference.
* Pointer can be used to make a function return more than one value simultaneously in an indirect manner.

→ When a function calls itself from within its body it is known as a recursive function.
→ A fresh set of variables are created every time a function gets called.
→ Understanding how a recursive function is working becomes easy if you make several copies of the same function on paper & then perform a dry run of the program.
→ Adding too many functions and calling them frequently may slow down the program execution.

(a) We can use different variation of the primary data type namely signed and unsigned char, long & short int, float, double and long double.

(b) The max value a variable can hold depends upon the number of bytes it occupies in memory.

(c) By default all the variables are signed. We can declare a variable as unsigned to accomodate bigger value by increasing the bytes occupied.

(d) We can make use of proper storage class like auto, register, static & extern to control four properties of a variable - storage, default initial value, scope & life.

Unit IV

ARRAYS :- Collection of items stored at contiguous memory location can elements can be accessed randomly using indices of an array.

Purpose -

* Consider a group of similar type item as a single unit
* Reduces the number of variables used
* Reduces no of lines of codes needed

Declaration & initialization

Syntax: <data_type> array_Name
Ex- int a[20]; float num[10], char name[20];

Initialization -

int a[5] = {10, 23, 15, 4, 90};
int a[] = {10, 23, 15, 4, 90};

Accessing and updating
→ An element is accessed by indexing the array name.
Syntax - array-name [index]

Command line arguments :-

→ Arguments that are passed to the C programs from the command line when they are executed.
→ These parameters are handled by Two arguments of main function namely
arg c: A integer value indicating the No. of argument p
arg v: A pointer array that points to the argument passe
to the progran

Page No.
Date

Structure :-

Need — Group related data of different data types and give a single name.

→ Collection of variables of different data types addressed with a single name.

→ User can define a structure as a defined data type with the help of typedef.

Declaration

Syntax

```
Struct Name
{
 <data type> member_name;
 <data type> member_name;
 - - - -
};
 list of variables;
```

Structure Pointer

```
#include <stdio.h>
struct student
{ int regNo.
  char name [25];
  int marks; } s;
int main () {
Struct student *stud;
Stud = &s;
```

```
scanf ("%d%s%d", &stud->regNo, stud->name, &stud->marks);
printf ("Register number = %d \n", stud->regNo);
printf ("Name %s \n", stud->name);
stud->marks = stud->marks + 10;
printf ("Marks = %d \n", stud->marks);
return 0;}
```

Union -

→ When we have to enter a large number of data, it will take a lot of time to enter them all.
→ When a program is terminated, the entire data is lost. Storing in a file will preserve the data even if the program terminates.
→ Having a file to hold all the data, will make it easy to access the contents of the file using a few commands.
→ It is also easy to move the data from one computer to another without any changes.

OPERATORS

Purpose :-

i) fopen - open a file - specify how it is opened.
ii) fclose - close an opened file
iii) fread - read from file
iv) fwrite - write to file
v) fseek / fSPtpos - move a file pointer to somewhere in file.

Page No. /
Date / /

Assignment

Unit I :-

① Differentiate : Machine learning , Assembly language and high level language

⟹ Machine language - It is the language written as string of binary 1's and 0's. It is the only language which a Computer understands without using a translation program.

Assembly language - It is a low level programming language that allows a user to write a program using alpha numeric mnemonic codes, instead of numeric codes. It is easier to remember and write than machine language.

High level language - It is a machine indipendent language. It enables a user to write programs in a language which resembles english words and familiar mathematical symbols. Each statement in a high level language is a micro instruction which is translated into several machine language instructions.

② Explain the keywords in C.

⟹ They are predifined, reserved words in C language each of which is associated with specific features. These words helps us to use the functionality of C language. They have special meaning to the Compilers. There

Page No.
Date / /

total 52 keywords in C.
eg - void, return, auto etc.

Unit II

(1) List the various operators in C with their precedence associativity and order of evaluation.

⇒ An operator is a symbol that tells the compiler to perform ~~the~~ specific mathematical or logical functions. Various operators are:-

i) Arithmatic operator - These operators performs mathematical operators like addition, substration, division, multiplication
ii) Relational operator - These are used to check relation b/w operands.
iii) Assignment operator - Used to assign value to a variable
iv) Logical operator - Used for decision making.
v) Conditional operator - The main purpose is in decision making statements. It is also known as Ternary operator.
vi) Bitwise operator - These are special operator for bit operation b/w two variables.

Category	operator	Associativity
Postfix	++ --	Left to right
Unary	+ - ! ~ ++ --	Right to left
multiplicative	* / %	left to right
Additive	+ -	left to right
Shift	<< >>	left to right
Relational	< <= > >=	left to right
Equality	== !=	left to right

Bitwise AND	&	left to right
Bitwise XOR	^	left to right
Bitwise OR	|	left to right
Logical AND	&&	left to right
Logical OR	||	left to right
Conditional	?:	right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	right to left
Comma	,	left to Right

(2) Write about the significance of Break and Continue with Sample programs.

→ Break – It is a jump statement which allows us to jump out of the loop without waiting to get back to the condition. The control automatically passes to the first statement after the loop. It is usually associated with an if statement.

Continue – The keyword continue allow us to take the control to the beginning of the loop. It is usually associated with an if.

Eg of Break –

```
#include <stdio.h>
int main()
{
```

```
int i;
for (i=100; i>=10; i--)
 {
 printf ("i : %d \n", i);
 if (i==99)
 {
 break;
 }
}
printf ("Out of loop");
return 0;
}
```

Output

```
i : 100
i : 99
Out of loop
```

Ex. of Continue

```
#include <stdio.h>
int main()
{
int i,j;
for (i=1; i<=2; i++)
 {
for (j=1; j<=2; j++)
 {
  if (i==j)
  Continue;
printf ("%d %d \n", i,j);
```

```
}
return 0;
}
```

Output

1 2
2 1

Unit III

(1) Explain function prototype, function difination, function calling using examples.

i) Function prototype – It is the declaration of function that specifies function's name, parameters and return type.

Syntax → return type function Name (type1 argument type2 argument);

ii) Function Definition – It ~~Consist~~ Contains the block of code to perform a specific task. When a function is called, the control of the program is transferred to the function difination and the compiler starts executing the codes inside the body of a function.

iii) Function Calling – It is called inside a program where it is required to call a function. It is only called by its name in the main() function of a program.

A:

```
#include <stdio.h>
Void message();  /* function prototype declaration */
int main()
{
 message();      /* function call */
 printf("Hello World");
 return 0;
}
Void message()  /* function definition */
{
printf("\n Good morning");
}
```

Output
Helloworld
Good Morning

Q Differentiate b/w call by value and call by reference with examples.

Call by Value	Call by reference
i) While calling a function, we pass values of variables to it.	i) While calling a function, we pass address (location) of variables to the function.
ii) The value of each variable in calling function is copied into corresponding dummy variables of the called function.	ii) The address of actual variables in the calling function are copied into the dummy variables of the called function.

iv) The changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function.

Eg
```
#include <stdio.h>
Void Swap (int x, int y);
int main()
{
int a = 10, b = 20
Swap (a, b);
printf ("a = %d b = %d \n", a, b);
return 0;
}
Void Swap (int x, int y)
{
~~void swap~~ {
int t;
t = x;
x = y;
y = t;
printf ("x = %d y = %d \n", x, y);
}
```

Output

x = 20 y = 10
a = 10 b = 20

iv) Using address we would have an access to the actual variables and hence we would be able to manipulate them.

```
#include <Stdio.h>
Void Swap (int *, int *)
int main()
{
int a = 10, b = 20;
Swap (&a, &b);
printf ("a = %d b = %d \n", a, b);
return 0;
}
Void swap (int *x, int *y)
{
int t;
t = *x;
*x = *y;
*y = *t;
printf ("x = %d y = %d \n", *x, *y);
}
```

Output
x = 20 y = 10
a = 20 b = 10

Page No. / Date / /

Unit IV

① Explain arrays and its types. Write an example program using multidimensional array.

⇒ An array is a linear data structure which is a collection of data items having similar data type stored in contiguous memory locations. The various types of arrays are:

i) One dimensional array – It is also called a single dimensional array where the elements will be accessed in sequential order.

ii) Multidimensional array – When the no. of dimensions specified is more than one, it is known as multi-dimensional array. It includes 2D & 3D arrays.

```
#include <stdio.h>
int main()
{
int stud [4][2];
 int i,j;
for (i=0; i<=3; i++)
 {
 printf ("Enter roll no. and marks");
 scanf ("%d %d", &stud[i][0], &stud[i][1]);
}
for(i=0; i<=3; i++)
 printf ("%d %d \n", stud[i][0], stud[i][1]);
}
return 0; }
```

(2) Describe about Command line argument.

⇒ Command line argument is a parameter supplied to the program when it is invoked. It is mostly used when we need to control our program from outside. These are passed to the main() method.

Syntax – int main (int argc, char *argv[])

Here argc counts the number of arguments on the command line argv[] is a pointer array which holds pointer of type char which points to the arguments passed to the program.

Ex:
```
#include <stdio.h>
#include <conio.h>
int main (int argc, char * argv[])
{
int i;
if (argc >= 2)
{
printf ("The arguments supplied are: \n");
for (i=1; i<argc; i++)
{
printf ("%s\t", argv[i]);
}
}
else
{
printf ("Argument list is empty: \n");
}
return 0;}
```

Page No. /
Date / /

Unit-④

① What is meant by Array of structures ? Give ex.

⇒ It is collection of different datatypes variables grouped together under a single name.

```
#include <stdio.h>
Void link();
int main();
{
struct book
{
Char name; float price; int pages;
};
struct book b[10];
int i; int dh;
for (i=0; i<=3; i++)
  {
    printf ("Enter Name, price & pages \n");
    scanf ("%c %f %d", &b[i].name, &b[i].price, &b
                                    [i].pages);
  while ((ch = getchar()) != '\n');
  }
for (i= 0; i<= 3; i++)
printf ("%c %f %d \n", b[i].name, b[i].price, b[i].
                                    pages);
return 0;
}
Void link()
{
```

```
float a=0, *b;
  b= &a;
  a= *b;
}
```

(2) Write about TYPEDEF in C.

⇒ typedef is a keyword used in C language to assign alternative names to existing datatypes. It is mostly used with user defined datatypes, when names of the datatypes becomes slightly complicated to use in programs.

Syntax -> typedef <existing_name> <alias_name>

Ex typedef unsigned long ulong;

The above statement defines a term ulong for an unsigned long datatype. Now this ulong identifier can be used to define unsigned long type variables

* Application of ~~datatypes~~ typedef:

i) With structures

```
typedef struct
{
   type member 1;
   type member 2;
   type member 3;
```

Page No. /
Date / /

};
type_name;

Here type_name represents the structure defination associated with it.

ii) With Pointers

- It can be used to give an alias name to pointers too

int *x, y;

By this statement, we disclose x as a pointer of type int, whereas y is a plain int variable.

typedef int* IntPtr;
IntPtr x, y, z;

By using typedef, we can declare any no of pointers in a single statement.