Date
Page

* What is a program?

ans A program is a set of instructions which is used to create a software program by using any one of the programming language. The program enables the computer to perform a particular operation

* What is programming language?

ans A specific language which is used to communicate with computer is called programming language. A language is a mode of communication, normally in programming context it is also called syntax. Programming language is write in specific language for example C, C++, Java, Python, etc.

Types of Programming language:

1) Low Level:

It is machine dependent (0 and 1). The processor runs low level programs directly without the need of interpreter. Because of this reason, it runs very fast.

Types of low level:

i) Machine level:

Machine language is also called machine code or object code. It is normally display in binary or hexadecimal form. It doesnot require a translator to convert the code or program.

ii) Assembly language:

It represent the set of instruction in a symbolic and human understanable form. It uses an assembler to convert the assembly language to machine language.

2) High level Language

It is design for developing user friendly software and websites. It requires a compiler or interpreter to translate the program into machine language. For e.g Python, Java, Javascript, PHP, C#, C++, COBOL, Pascal, FORTRAN, Swift Programming language.

* What is error?

Error are the problems or faults that occur in programming. The error are detected during the time of compilation or excecutation. It must be removed from the program for successful execution of program.

Types of error:

i) Syntax Error

A syntax error is an error in the syntax of a sequence of characters that is intended to be written in a particular programming language. This kind of error are generally indicated by the compiler before compilation. Sometimes these are known as compile time error. eg??

ii) Runtime Error

A runtime error in a program is an error that occur while the program is running after being successfully compiled. This kind of error are occurred, when the program is executing. As this is not compilation error, so the compilation will be successfully done. We can check this error if we try to divide a number with 0.

iii) Logical Error

A logical error is a mistake in a program's source code that results in incorrect or unexpected behaviour. It occour when there is a fault in the logic or structure of the problem. Logical error do not usually cause a program to

crash. However, logic error can cause a program to produce unexpected results.

1) Example:

```
# include <stdio.h>
main(){
    printf("Hello World")
}
```

Output: Error] expected ';' before '}' token

2)

```
# Include <stdio.h>
main(){
    int x = 52;
    int y = 0;
    printf("Div : %f", x/y);
}
```

Output: Program crashes during runtime

3)

```
# include <stdio.h>
main(){
    int i;
    for(i = 0; i < 5; i++); {
    printf("Hello world");
    }
}
```

Output: Here we want the line will be printed five times. But only one time it will be printed for the block of code.

* Characteristics of good Programming Language.

1) Evolution and Update

Adaptibility to evolving the industry trade is key. Good programming languages under goes various update and enhancement to stay relevant.

2) Clarity and Readability

A good programming language should have clear syntax and instruction that are easy to understand. It reduces the chances of error by enhancing collaboration by developers.

3) Flexibility and Versability

A good programming language should be flexible enough to adapt different paradigms. It allows developers to use it for a wide range of project.

4) Scalability

Good programming language should facilated the development of scalable solution and ensuring they can handle increase work load without compromising performance

5) Error Handling

Good programming language provide robust mechanism for error detection and recovery.

* Software Development Model /~~Software development life cycle (SDLC)~~ → /Process

Software Development model are various process or methods that are choosen for various development depending upon objectives and goal of the projects.

Types:

1) Waterfall Model:

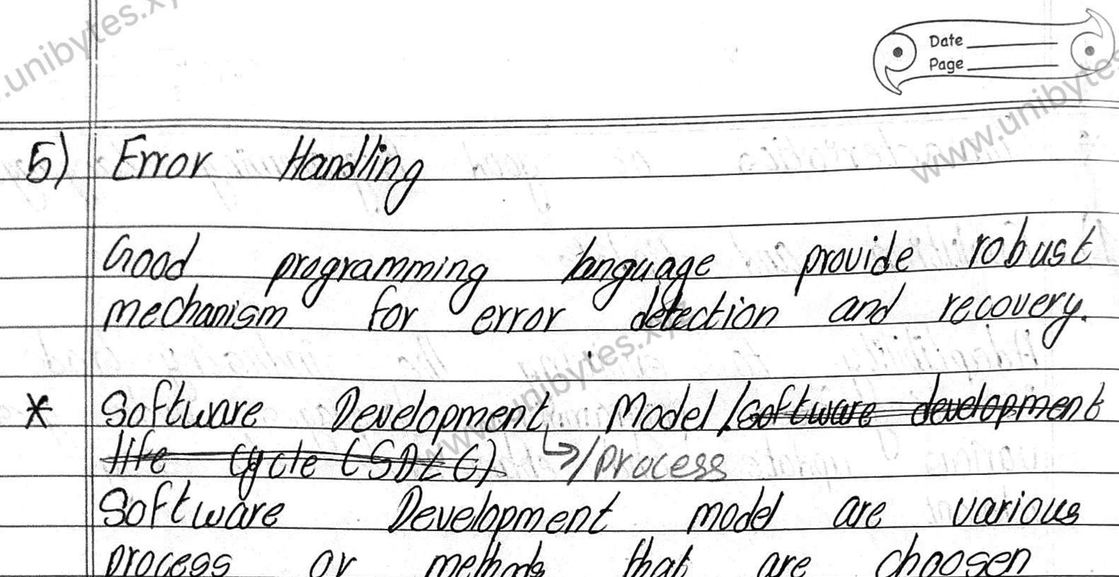


Fig: Waterfall Model.

i) Requirement

This is the first phase where requirement of system are collected and documented

Based on this document design is build.

ii) Design

Design phase helps to prepare the blue print of the software. It helps to detect the problems before hand. Design of interface number of users are calculated and based on this calculation design is completed.

iii) Implementation

In implementation phase hardware, software and application programs are installed and database design is also implemented. Actual coding is also done in this phase. This is the longest phase in water fall model because the software has to go through testing, coding and debugging process.

iv) Verification

In this phase the software is verified and evaluated that have created the right product. Various types of testing are done and every ~~one~~ area of software is test. This type of verification helps to reduce, the risk of software failure.

v) Maintanance

This is the last phase where we solve the problems if any error occurs in the software. Taking care of finished software and maintaining it as per time is called maintanance.

2) Incremental Model.

Incremental Model is process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each modules goes through the requirements, design, implementation and testing phases. Every subsequent release of the modules adds function to the previous release. The process continues until the complete system achieved.

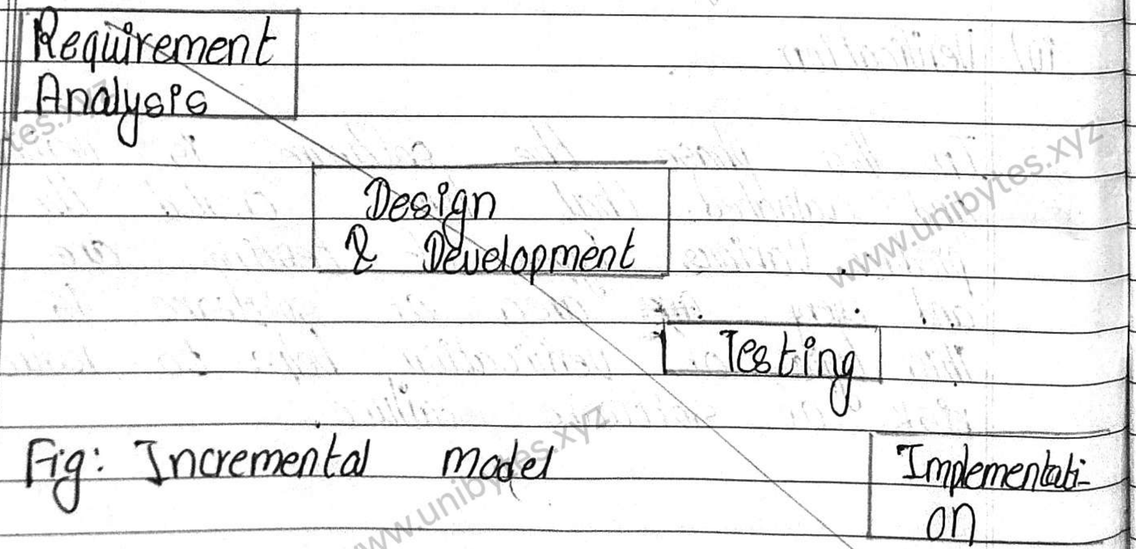


Fig: Incremental model

1) Requirement analysis:
In the first phase of the incremental model, the product analysis expertise identifies the requirements. are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

ii) Design and Development:
In this phase, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and developmental phase.

iii) Testing
This phase check the performance of each existing function as well as addition and functionality. In the testing phase, the various methods are used to test the behavior of each task.

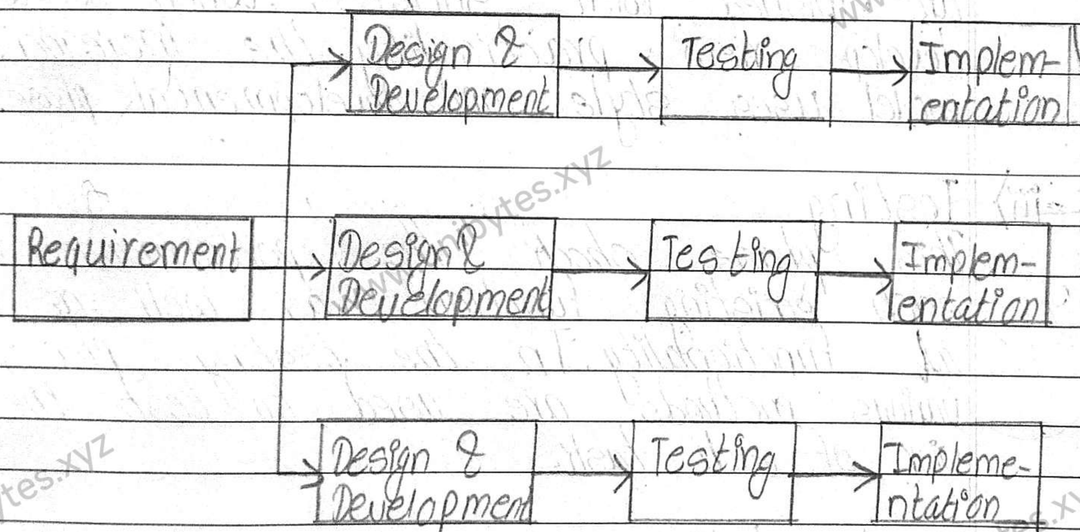
iv) Implementation
This phase enables the coding phase of the development system. It involves the final coding that design in designing and development phase and tests the functionality in testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product.

* Advantages

i) Error are easy to be recognized.
ii) Easier to test and debug.
iii) More flexible.
iv) The client gets important functionality early.

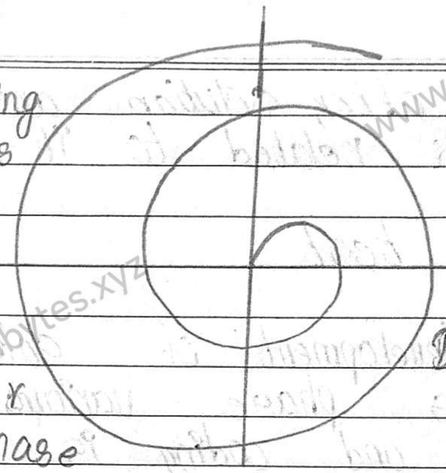
* Disadvantages

i) Need for good planning
ii) Total Cost is high
iii) Well defined module interfaces are needed.



3) Spiral Model

This model is the mixture iterative and waterfall model. It is used in large and complex project. It is named after its loop. Because look like a spiral in which



a long curve line start from the centre point and makes ~~it~~ many loops around it. The number of spiral is not decided in advance. The software project goes through this loop again and again iteration.

It is also called meta model, which was first develop by Barry Boehm in 1988. There are four phases in spiral model.

i) Determining Objective

In the first phase the requirements are connected so that the objectives are analysed and identified on the basis of ~~requir~~ collected data alternative solution are also purposed.

ii) Identifying and resolving risks.

In this phase all the purposed solution assessed and the best solution

is created. There after, solution are analysis and the risk related to it are identified.

iii) Develop and test

Now the development of software started. In this phase various features are implemented and coding is done.

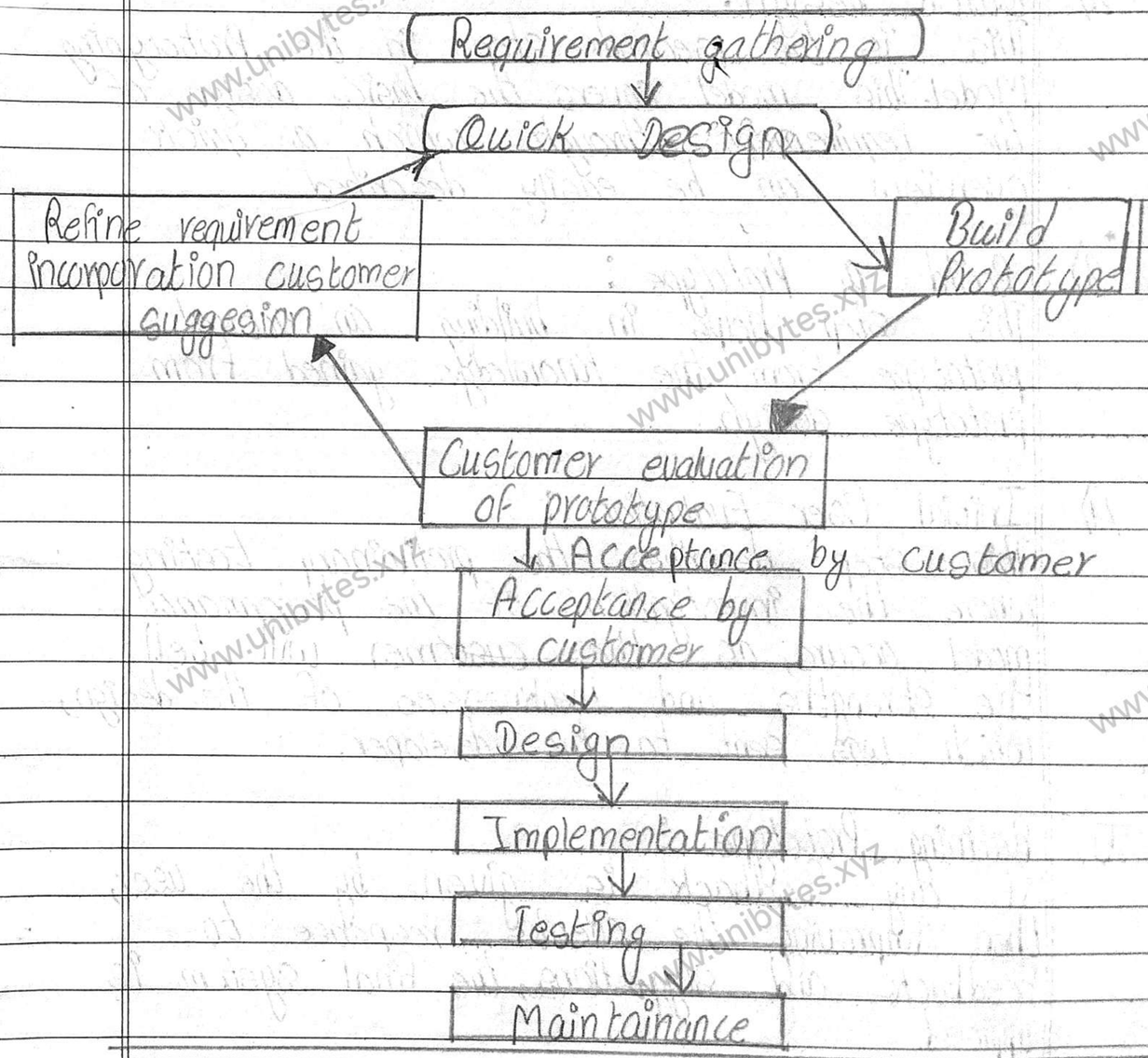
iv) Review plan for the next phase

In this phase the developed version of the software is given to the customer and he evaluate it. The customer again gives feedback and tells new requirements. Finally planing of the next phase is started.

4) Prototype Model

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small-scale facsimile of the end product and is used for obtaining customer feedback.

The prototype Model is one of the most popularly used SDLC. This model is used when the customers do not known the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for typi developing the final product.



1) Requirement Gathering and Analysis:

This is the initial step in designing a

prototype model. In this phase, users are asked about what they expect or what they want from the system.

2) Quick Design:
This is the second step in the Prototyping Model. This model covers the basic design of the requirement through which a quick overview can be easily described.

3) Build a Prototype:
This step helps in building an actual prototype from the knowledge gained from prototype design.

4) Initial User Evaluation:
This step describes the preliminary testing where the investigation of the performance model occurs, as the customer will tell the strengths and weaknesses of the design, which was sent to the developer.

5) Refining Prototype:
If any feedback is given by the user, then improving the client's response to feedback and suggestions, the final system is approved.

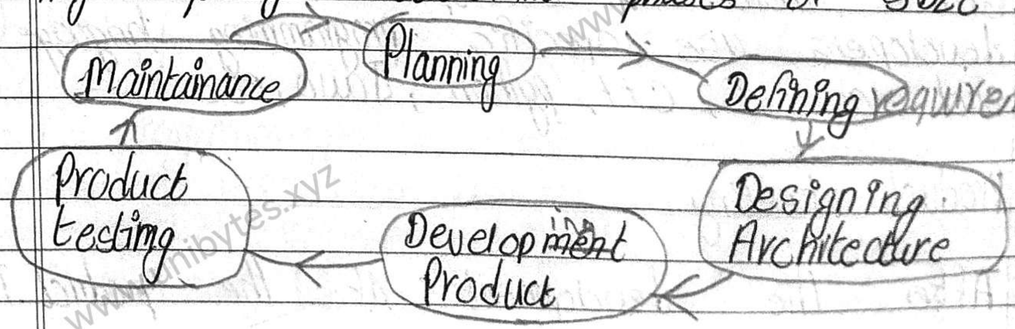
6) Implementation:
~~This is final step in~~
During this stage, the software design is realized as a set of programs and program units.

7) Testing:
Once the program modules are ready, each of the program modules is tested independently as per the specification of the user and debugged.

8) Operation and Maintainance:
In this stage, the system is installed and put into practical use. Maintenance involves correcting and improving the implementation of system units and enhancing the system's services as new requirement are discovered.

* Software Development ~~Models~~ Lifecycle (SDLC)

SDLC is structured process that is used to design, develop and test good quality software. It is a methodology that defines the entire procedure of the software development step by step. The goal of SDLC is to deliver high quality software. The phases of SDLC:



i) Planning

Planning is attained from custom of input. The basic design is produce at this level. Requirement analysis is also perform in this stage.

ii) Defining Requirement :

In this stage all the requirement for the software is specified. These requirements are approved from customer, market analyst and stack holder.

iii) Designing

With the requirement collected in the previous stage multiple design of the product are created and after evaluation most pratical and logical design is choosen for the development.

iv) Developing Product

The fundamental development of the product starts here. For development of software developers use specific programming language such as C, C++, Python, Java, etc.

v) Product testing

After the development of the product. Testing of the software is necessary to ensure it smooth excecution. Therefore, at this stage all the programmer faults are ~~fin~~ track, fixed and re-tested.

vi) Maintainance

After product testing the conclusion product is released. It is then tested real in the industry environment. It is importedant to ensure its smooth performance.

*1) What is the history of C programming
2) What are the features of C programming.

3) What is Iteration?
ans This is the process of repeating the set of operations or steps. It is like doing some ~~thing~~ thing over and over

4) What is recursion.
ans The process in which the function call itself is called recursion. It is needed to reduce the length of the code.

1) History of C:

C programming is a general purpose, procedure-oriented programming language. It is both machine independent language developed by Dennis Ritchie in the early 1970s. It is now one of the most

exact date is 1972 A.D

popular and influential programming language worldwide.

C is popular for its simplicity, efficiency and versatility. It has powerful features including low-level memory access, a rich set of operators, and a modular framework. The languages that are influenced by C include Java, PHP, Javascript, C#, Python and many more. These languages have designed their syntax, control structures and other basic feature from C.

C support different hardware and operating system due to it portability. Dennis Ritchie created C at Bell Laboratories in the early 1970s. It developed from an older language named B that Ken Thompson created. The main purpose of C's creation was to construct the Unix operating system, which was crucial in the advancement of contemporary computer.

The feature of C language are :-

i) C is a General-Purpose Language

The C language hasn't been developed with a specific area of application as a target. From system programming to photo editing software, the C programming language is used in various applications. Some of the common application of C programming include the development of Operating systems, databases, device drivers, etc.

ii) C is Portable

C programs are machine independent which means that you can compile and run the same code on various machines with none or some machine-specific changes.

It provides the functionality of using a single code on multiple systems depending on the requirement.

iii) C is Extensible

It is an extensible language. It means if a code is already written, you can add new features to it with a few alterations.

iv) Recursion in C

Recursion in C programming provides the functionality of code reusability and backtraking

v) Rich set of built-in Operators

It is a diversified language with a rich set of built-in operators which are used in writing complex or simplified C programs.

vi) Libraries with Rich Functions

Robust libraries and functions in C help even a beginners coder to code with ease.

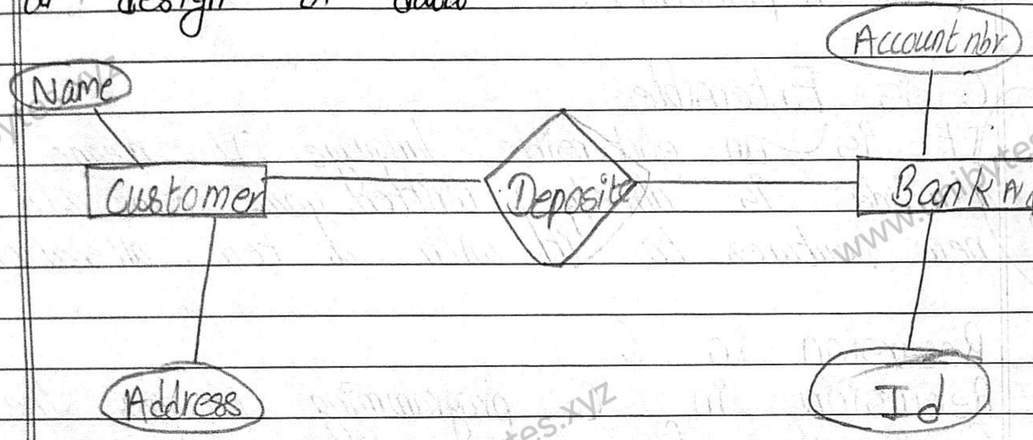
vii) Middle-Level Language

As it is a middle-level language so it has the combined form of both capabilities of assembly language and features of the high-level language.

System Development ~~Software~~ tools

i) E-R Model (Relational)

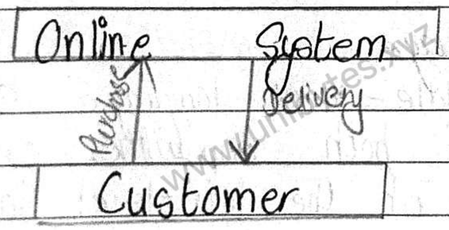
E-R relationship model is an database model base on the notation of real world entities. It is used for the conceptual design of data



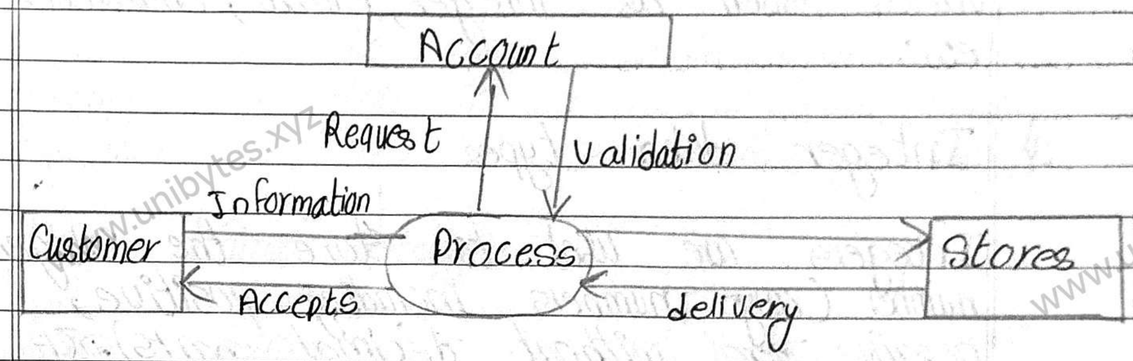
There are various mapping cardinalities.

ii) DFD (Data Flow Diagram)

It is a graphical representation of flow of information. It is capable of explaining incoming data flow, out going data flow and stored data. For e.g



E.g. 2:



iii) Decision Table

It represents condition and the respective action to be taken to address them in a structure tabular format. Basic steps to create decision table:

a) Identify all possible condition to be address.
b) Determines action for all identified conditions.
c) Create maximum possible rules.
d) Define action for each rules

* Data Types

It specifics the type of data that the variable can store. Each data types required different amount of memory and has some specific operation which can be performed over it.

Types:

i) Primitive Data Types
Primitives are the most basic datatype

that are use for representing simple values such as integer, float, characters, etc.

a) Integer data types

Integers are used to store the integers number (any numbers including positive, negative, zero (without decimal parts). It occupies 4 byte in memory. The format specifier is "%d". It is denoted by int.

e.g1 int a;
a=5

/ int b;
b=5.5; (incorrect)

/ int c;
c="apple" (x)

b) Float

It is used to store floating datatype (store decimal values). The memory need to store floating value is 4 byte. The format specifier for float is "%f". It is represented by float.

e.g:
float a;

a=5.5 ✓ / a="apple" x

8 bit = 1 byte

c) Character

It allows to store single character. It occupies 1 byte in a memory. The format specifier is "%c". It is denoted by Char

ii) User defined datatype

Users defines this type of data himself. For e.g: structure, union. In this data type various other data type like integer, float, character, boolean, etc are combined to form a different data types.

e.g
```
Struct student {
    int roll;
    float marks;
    char name[10];
} annapurna[25];
```

iii) Derived Data Type

The data type that are derived from the primitive data type are referred as derived data types. For e.g: function, array, pointer, etc.

```
int no_of_students[35]
```

H.W a) Write the general overview of C programs.

To learn C effectively, we need to understand its structure first. A typical

Structure of a C program includes several parts. The following steps show the C structure of a regular C program

1) Include header files

Include necessary header files that contain declarations of function, constants, and macros that can be used in one or more source code files. Some popular header files are:

i) Stdio.h
It provides input and output functions like printf and scanf.
```
#include <stdio.h>
```

ii) math.h
Includes mathematical functions like sqrt, sin, cos, etc.
```
#include <math.h>
```

iii) String.h
Includes function for manipulating strings, such as strcpy, strlen, etc
```
#include <string.h>
```

iv) time.h
contains function for working with date and time.
```
#include <time.h>
```

v) limits.h
Defines various implementation specific limits on integer types.
```
#include <limits.h>
```

* Global Declarations in C
They are optional:
```
int globalVariable;
void sampleFunction();
```

Declare global variables and functions that will be used across different parts of the program. Take a look at following e.g

```
#include <stdio.h>
int globalVariable;
int main()
{
  // Rest of program
  return 0;
}
```

2) Main Function

Every C program must have a main function. It is the entry point of the program.
Eg
```
Int main() {
  float radius = 5.0
  float area = PI * radius * radius;
  printf("Area of the circle: %f \n", area)
  } return 0;
```

3) Functions in C

Define other function as needed. The main function may call these functions. E.g:

```
# include <stdio.h>
void sampleFunction ();
int main () {
    ~~retur~~ // Programming statements
    return 0;
}
// Global function definition
void sampleFunction () {
// Function Programming statement implementation
}
```

A C program can vary from 3 lines to millions of lines and it should be written into one or more text files with extension ".c", for e.g: hello.c. You can use "vi", "vim" or any other text editor to write your C program into a file.

~~* Structure of C~~

* Structure of C program

comment → // This program explain the addition
// of two integer number

Link section →
```
# include <stdio.h>
# include <math.h>
# include <conio.h>
```

Global declaration section →
```
define float 9.8
```

main function section →
```
main () {
    add ()
    sub ()
}
```

Functions
```
add () {
}
sub () {
}
```

* Printf
It is used to display the data.
For e.g
```
printf ("Hello World");
```

* WAP to display Hello World by using C.

```
// The program show hello world
#include<stdio.h>
void main()
{
  printf("Hello World");
}
```

* WAP to display your detail

```
// The program shows my detail
#include <stdio.h>
void main()
{
   printf("My name is Bikram Gyawali\n");
   printf("I am from Butwal\n");
   printf("I read in BCA 2nd sem");
   return 0;
}
```

* Write short notes on Key words, identifier, modular programming, structure programming.

The character set that is collected by the compiler to syntactic unit is called tokens that is the basic syntactic unit of C program.

* Keywords:

Keywords are those predefined words that have special meaning in the compiler and they cannot be used for any other purpose. As per the C99 standard, C language has 32 keywords, such as: int, float, char, double, etc. C is a sensitive language. Hence, int is a keyword but INT, inT are not recongniz-ed as a keyword.

* Identifier

Identifier are the user-defined name given to make it easy to refer to the memory. It is also used to define various elements in the program, such as function, user-defined type, labels, etc. It helps in identifying variable, constant, function, etc. It represent the name in C. first letter must be alphabet or underscore.

* Modular Programming

Modular Programming is a general programming concept where developers seprate program functions into independent pieces. A module is a separate software component. It can often be used in a variety of applications and functions with other component of the system.

* Structure Programming

C is a structure programming because it divides a large problem into smaller modules called function or procedures, each of which handles a specific responsibility. A collection of such functions constitutes the program that solves the entire problem.

* WAP to produce following pattern.

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

// This program is to produce the pattern

```
# include <stdio.h>
main(void) {
    printf("* * * * \n");
    printf("* * * \n");
    printf("* * \n");
    printf("* \n");
}
```

* WAP to add any two number

// Program to find the sum

```
# include <stdio.h>
main() {
    int a = 5;
    int b = 6; // initialization
    int sum; // declaration
    sum = a + b;
    printf("Sum of 5 and 6 is %d", sum);
}
```

* WAP to subtraction, multiplication and division. two numbers;

// Program to subtract, multiply and divide

```
# include <stdio.h>
main() {
    int a = 10;
    int b = 5;
    int sub;
    int mult;
    int div;
    sub = a - b;
    mult = a * b;
    div = a / b;
    printf("Substraction is %d\n", sub);
    printf("Multiplication is %d\n", mult);
    printf("Division is %d\n", div);
}
```

* WAP to calculate the Simple interest:

// Calculating interest

```
#include <stdio.h>
main () {
  float p, t, r, si;
    p = 4000;
    r = 2.2;
    t = 2;
  si = (p*t*r)/100;
  printf("The Simple interest is %f, si);
    }
```

* WAP to find the cube of a number taken from user.

// Cube of a number

```
# include <stdio. h>
main () {
    int a, cube;
  printf (" Give the number %d \n", a);
  Scanf(" %d" & a);
   cube = a * a * a;
  printf (" The cube is %d \n", cube);
    }
```

1* WAP to find area and perimeter of a rectangle.

2. WAP to find the area between two co-centric circle

1) // Area and Perimeter of rectangle.

```
# include <stdio.h>
main () {
  float l, b, area, per;
  printf("Enter the length \n", l);
   Scanf(" %f", & l);
  printf("Enter the bredth \n", b);
   Scanf(" %f", & b);
    area = l * b;
    per = 2 * (l + b);
  printf (" The area is %.2f \n", area);
  printf (" The perimeter is %.2f \n", per);
    }
```

2) // The co-centric Circle

```
# include <stdio.h>
main () {
{
  int r1, r2;
  float a1, a2, a3;
  printf ("Enter radius of big circle: \n");
  Scanf(" %d ", & r1);
  printf ("Enter radius of small circle: \n");
```

```
scanf("%d", &r2);
a1 = 3.14 * (r1 * r1);
a2 = 3.14 * (r2 * r2);
a3 = a1 - a2;
printf("\n Area between concentric
    circle : %.2F", a3);
}
```

Homework:

1) WAP that calculates the volume of a sphere.
2) WAP that ~~inputs~~ prints the perimeter of a rectangle using its height and width as inputs.
3) Write a C program that converts kilometer per hour to miles per hour.
4) To takes hours and minutes as input and calculate the total hour.

1) //program to calculate volume of a sphere.

```
#include <stdio.h>
main (){
    float r, v, pi;
    pi = 3.14;
    printf("Enter the Radius :", r);
    scanf("%f", &r);
    v = (4/3) * pi * (r * r * r);
    printf("The volume of a sphere : %.2f", v);
}
```

2. //Program to calculate perimeter of a rectangle using height and width

```
# include <stdio.h>
main () {
 float h, w, p;
 printf("Enter the Height :", h);
 scanf("%f", &h);
 printf("Enter the Width :", w);
 scanf("%f", &w);
 p = 2 * (h + w);
 printf("The perimeter of a rectangle : %.2f", p);
}
```

3) // Program to calculate kilometer per hour to miles per hour.

```
#include <stdio.h>
main (){
  float kph, c, mph;
  c = 0.62137119;
  printf("Enter the kilometer per hours:", kph);
  scanf("%f", &kph);
  mph = kph * c;
  printf("Miles per hour : %.2f", mph);
}
```

4) // Program to calculate total number of minute.

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

```
int h, c, m;
printf("Enter the hours :", h);
scanf("%d", &h);
printf("Enter the minute :", m);
scanf("%d", &m);
c = (h*60) + m;
printf("Total number of Minute : %d", c);
}
```

5) //program to calculate third angle

```
#include <stdio.h>
main() {
  int a, b, c, t;
  t = 180; // t means total area of a triangle
  printf("Enter the first Angle :", a);
  scanf("%d", &a);
  printf("Enter the second Angle :", b);
  scanf("%d", &b);
  c = t - (a+b);
  printf("Value of Third Angle : %d", c);
}
```

Conditional Statement

- Conditional statement are used to evaluate one or more conditions make the decisions whether to execute the statement or not.
- Conditional statement in programming language decide the direction of the flow of programs.

++x, Prefix first add the value and show in result
e.g y= x=10 y=++x then x=11 & y=11
x++ postfix, first show the result and add in the number
e.g x=10, y=x++, x=11, y=10

Types of conditional statement

i) if
ii) if ____ else statement
iii) else if
iv) nested if

i) if

- The if statement is used to define whether a certain block of statements will be execute or not.
- If the condition is true, then the blocks of statements are executed otherwise. It is not executed.

Syntax:

```
If (money > 100)
{
  printf("I will eat momo");
}
```

1)
```
#include <stdio.h>
#include <conio.h>
main () {
    int m;
    printf(" Enter the amount of money ", m);
    scanf(" %d ", &m);
    if (m >= 4500){
        printf (" You can travel by plan ");
    }
    else {
        printf (" Sorry increase your budget ");
    }
}
```

2) WAP to find the number odd or even

// Program to find the odd or even

```
# include <stdio.h>
main () {
        int n;
    printf (" Enter the number ", n);
    scanf (" %d ", &n);
    If (n % 2 == 0) {
        printf (" The number is even ");
    }
    else {
        printf (" The number is odd ");
    }
}
```

3) WAP to swap two number.

// Swap 2 number

```
#include <stdio.h>
# include < conio.h>
main () {
    int n1, n2, temp;
    printf (" Enter the first number ");
    scanf (" %d", &n1);
    printf (" Enter the second number ");
    scanf (" %d", &n2);
    temp = n1;
    n1 = n2;
    n2 = temp;
    printf (" Swapped values, number 1 = %d number 2 = %d", n1, n2);
}
```

4) # WAP to take input of CP and SP and determine whether it is gain or loss

```
# include <stdio.h>
# include <conio.h>
main () {
        int CP, SP;
printf("Enter the Cost price");
scanf("%d", &CP);
printf("Enter the Selling price");
scanf("%d", &SP);
If(SP > CP){
        printf("You get profit"); }
else {
        printf("You are in loss");
}
}
```

Nested If

If one if-else contains another if-else it called nested if-else. It is used when series of decission are to be taken.

Syntax

```
If (condition) {
        ↓
else If (condition) {
        }
        else {
        } }
```

```
else {
}
```

5) WAP to find whether a person is eligible to vote or not.

```
# include <stdio.h>
# include <conio.h>
main () {
        int age;
printf("Enter the age of a person");
scanf("%d", &age);
if (age >= 18) {
        printf("You can vote"); }
else {
        printf("You can't vote");
}
}
```

6) WAP to find ~~whethe~~ if the person is over 90 he must carry wheel chair and if he is over than 18 he can vote otherwise he can't vote.

7) WAP to find the greater and smallest among there three no.

8) WAP to read the two variable and swap them without using third variable.

9) What is if else ladder? Explain with flow chart.

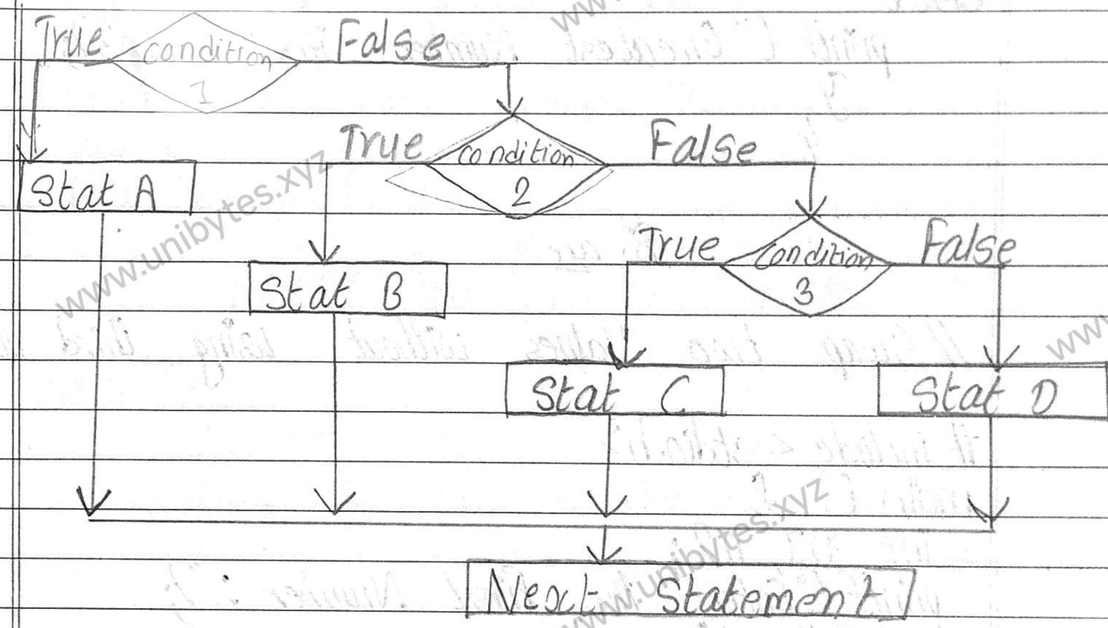
Q.N 9

If else if ladder in C programming is used to test a series of conditions sequentially. This is a type of nesting in which there is an if-else statement in every else part except the last else part.

If any of the conditional expressions evaluate to be true, the appropriate code block will executed, and the entire if-else ladder will be terminated. The general syntax for the if-else-if ladder is follows:

```
if (condition 1)
 Stat - A;
else if (condition 2)
stat - B;
else if (condition 3)
stat - C;
...
else
next statement;
```

The condition are evaluated from top downward.



7) // Greatest number among three numbers

```
#include <stdio.h>
main ( ) {
        int n1, n2, n3;
 printf ("Enter the first Number : ");
 scanf ("%d", &n1);
 printf ("Enter the Second Number : ");
 scanf ("%d", &n2);
 printf ("Enter the Third Number : ");
 scanf ("%d", &n3);
 if (n1>n2 && n1>n3) {
 printf ("Greatest Number is %d", n1);
 }
else if (n2>n1 &&n2>n3) {
 printf ("Greatest Number is %d", n2);
 }
```

```
else{
    printf("Greatest Number is %d", n3);
  }
}
```

8. aa

// Swap two Values without using third variable

```
#include <stdio.h>
main() {
  int n1, n2;
  printf("Enter the First Number :");
  scanf("%d", &n1);
  printf("Enter the second Number :");
  scanf("%d", &n2);
  n1 = n1 + n2;
  n2 = n1 - n2;
  n1 = n1 - n2;
  printf("After Swapping, n1 = %d and n2 = %d\n",
    n1, n2);
}
```

10) WAP to find the gain and loss with amount.

// Program to find gain and loss

```
#include <stdio.h>
main() {
  int sp, cp, d, l;
  printf("Enter the cost price :");
  scanf("%d", &cp);
  printf("Enter the Selling price :");
  scanf("%d", &sp);
  if (sp > cp) {
      d = sp - cp
      printf("You gain amount Rs %d", d);
  }
  else {
      l = cp - sp;
    printf("You are in loss Rs %d", l);
  }
}
```

6. aa

```
#include <stdio.h>
main() {
  int a;
  printf("Enter the age of the person:");
  scanf("%d", &a);
  if (a > 18) {
    if (a > 90) {
      printf("Please carry Wheelchair with you");
    }
    else{
        printf("You can vote");
    }
  }
  else {
    printf("You can't vote");
  }
}
```

What is switch?

The switch statement ~~ste~~ select one of the many options. The switch expression is evaluated one. The value of the expression is compare with the value of each case. The "break" statement breaks out of the switch block and stop execution.

Syntax:

```
Switch~~Case~~ (value) {
    case 1 : Expression;
             break;
    case 2 : Expression;
             break;
    case 3 : Expression;
             break;
    Default : Exp;
}
```

Operator in C

1. Arthemetic operator
2. ~~Conditional operator~~ / Logical operator
3. Bitwise Operator
4. Assigment operator

The arthemetic operator are used to perform ~~alt~~ arthemetic operations on operand. There are various arthemetic

Operator in C

a) Pluse = (A+B)
b) - = (A-B)
c) multiply = (A*B)
d) Divide (/) = (A/B)
e) % = (A%B) [Modulos]
f) Uniary = (+a) [working as a = a+1]
g) Uniary - = (-a)

2) Logical Operator

They are used to combine two or more conditions. The result of the operation is true or false. It is denoted by following symbols.

i) && - logical And a&&b
ii) || - logical OR a||b
iii) ! - logical NOT !a

3) Assigment Operator

They are used to assign value to a variable. List of assigment operator.

	Symbol	Description	example
i)	=	assigne the value	a=5;
ii)	+=	add the right operand and left operand and assign the value to left operand	a+=b

iii) -=
iv) *=
v) /=
vi) %=

3. Bitwise Operator

They are used to perform bit-level operations on integer data types. The bit wise operator is a type of operator that operates on bit arrays, bit strings, and adjust binary values with individual bits of the bit level.

For example, a logical AND (&) of each bit pair result in a 1 if both the first AND second bits are 1. If only one bit is a 1, the result is 0.

- Bitwise operator in C

& - AND
| - OR
~ = Not (complement)
<< = left shift
>> = Right Shift
^ = Exclusive OR

4) Conditional operator

It is also known as a ternary operator. The conditional statements are the decision-making statements which depends upon the output of the expression. It is represented by two symbols, i.e ? and :

The behaviour of the conditional operator is similar to the 'if-else' statement. It works on three operands, so it is known as the ternary operator.

Syntax:

Exp 1 ? Exp 2 : Exp 3;

* Increment Operator

Increment operator are the operator of the C programming language used to increase the given variable's value by 1. The incremental operator is represented as the double plus (++) symbol, which means the value is increment by 1.

Types:

i) Pre-Increment Operator

The pre-increment operator is represented used to increment the value of an operand by 1 before using it in the mathematical expression. In other words, the value of a variable is first incremented, and then the update value is used

In the expression.

Syntax:

```
x = ++a;
```

// Program for using pre-increment;

```
#include <stdio.h>
main () {
  int a = 7, b=2;
  b = ++a;
  printf("The value of a is %d ",a);
  printf("The value of b is %d", b);
}
```

output:

The value of a is 8
The value of b is 10

ii) Post - Increment Operator

It is an increment operator, represented as the double plus (a++) symbol followed by an operator 'a'. It increment the value of operand by 1 after using in the mathematical expression. In other words, the variable's original value is used in the expression first, and then the post-increment operator updates the operand value by 1.

Syntax:

```
x = a++;
```

// Using post - increment operator.

```
#include <stdio.h>
main () {
  int a=7, b=0;
  b= a++;
  printf("The value of a is %d \n", a);
  printf("The value of b is %d", b);
}
```

Output:

The value of a is 8
The value of b is 7

* ~~#~~ Size of Operator

This is uniary operator which finds at number of byte that any object occupy in a computer memory. The syntax is

~~size of~~ sizeof (datatype);
sizeof(int);

* WAP to take three different number and find the middle value.

```
#include <stdio.h>
main () {
    int a, b, c, m;
  printf("Enter three different number :");
  scanf("%d %d %d", &a, &b, &c);
  if((a>b && a<c) || (a>c && a<b)){
  m = a; }
```

```
else if ((b>a && b<c) || (b>c && b<a)){
    m=b;
} else{
    m=c; }
printf("The middle value is %d", m);
}
```

Ques
20/10/10

* LOOP

Loop is used to repeat a block of code until the specifing condition is meet. Loop consist:

i) Initalization
ii) Condition
iii) Increament / Decrement

There are three types of loop:

i) For loop
ii) While loop
iii) Do-While loop

i) 1

i) For loop

The syntax for the for loop is

```
for (i=0; i<=10; i++)
{
  // Statement
}
```

1) # WAP to find the sum of integer number 1000 to 2000.
2) WAP to display only odd numbers from 1 to 100

1) // Program to find the sum of integer number 1000 to 2000.

```
#include <stdio.h>
main(){
 int i, s=0;
 for(i=1000; i<=2000; i++){
  s+=i; // adding i with s and store the value ins
 }
 printf("The sum is %d", s);
}
```

output:

The sum is 1501500

2) // To find the odd number.

```
#include <stdio.h>
```

```
main(){
   int i;
  for(i=0; i<=100; i++){
  if(i%2!=0){
  printf("%d", i);
  }
  else{
    printf("\n");
  }}}
```

3) // To print even number from 1 to 100

```
#include<stdio.h>
main(){
   int i, r;
  for(i=0; i<=100; i++){
   r=(i%2==0)?printf("%d\n",i):printf(" ");
  }}
```

While loop

Syntax:

```
                  → // print // to print 10 put i=1;
    i=0; //Initialize
    while(i<=10){ → // condition
          //statement
          i++; //increment & decrement
    }
```

4) WAP to print 1 to 9 using while loop.

```
#include<stdio.h>
main(){
   int i=0;
  while(i<10){
        printf("%d", i);
        i++;}
  }
```

Homework:

1) WAP to get the sum from 1 to 10 using while-loop.
2) WAP to calculate the products of numbers from 1 to 5 using while-loop.
3) WAP that prompts the use to enter a positive integer. It then calculate and print the factorial of that number using a while loop.
4) WAP to check if a given number is palimdrone using while-loop.

1. ans

```
#include<stdio.h>
main(){
    int i=1, sum=0;
    while(i<=10){
        sum+=i;
        i++;
    }
    printf("Sum of Numbers: %d", sum);
    }
```

Note: while maa 0 bata start gare loss then yakhi natra 1 badi print hunxa.

3.ans for i=0, i<=5
↳ 6 time print hunxa

```
#include <stdio.h>
main(){
  int i=1, n, factorial=1
  printf("Enter the Number:");
  scanf("%d", &n);
  while(i<=n){
     factorial *= i;
      i++;
  }
  printf("The factorial of %d is: %d", n, factorial);
   }
```

2.ans

```
#include <stdio.h>
main(){
   int i=10, n=5, result=1;
   while(i<=5){
        result *= i;
        i++;
      }
printf("The multiplication of 1 to 5 is: %d",
      result); }
```

4.ans

```
#include<stdio.h>
main(){
   int n, rev=0, org, rem;
// rev=reverse, org=Orginal, rem=reminder
printf("Enter a number: ");
scanf("%d", &n);
org=n;
while(n!=0){
      rem = n%10;
      rev = rev*10+rem;
      n/=10;
}
/* Using conditional statement
   if(org==rev){
        printf("%d is Palindrome", org); }
else{
    printf("%d is not Palindrome", org);
   } */
// Using conditional Operator
   (org==rev)? printf("%d is Palindrome",
org): printf("%d is not Palindrome", org);
}
```

* Do-While loop

It is used when at least one statement has to be excuded.

Syntax:

```
do{
   // statement
   ++ increment / decreasment
} while(// condition)
```

* 5 hello world using while.

```
#include <stdio.h>
    main()
    {
      int i=1;
    while (i<=5){
            printf("Hello world!\n");
        }
        }
```

* Different between for, while, do-while loop.

Chapter 4

#) What is an array?

- The collection of similar data types is called array.
- It is used in programming language to write syntax clearly.
- It helps to reduce the error in program.

Syntax:

- int roll[10];

1) WAP to add 4 number ~~to add~~ using array.

```
#include <stdio.h>
main{
    int n[4],sum;
printf("Enter first number :");
 scanf("%d",&n[0]);
printf("Enter Second number :");
 scanf("%d",&n[1]);
printf("Enter Third number :");
 scanf("%d",&n[2]);
 printf("Enter fourth number :");
 scanf("%d",&n[3]);
 sum=n[0]+n[1]+n[2]+n[3];
 printf("The sum is %d",sum);
}
```

Using array and loop

```
#include <stdio.h>
main(){
    int n[4], sum, i, sum=0;
    for(i=0; i<4; i++)
    {
      printf("Enter no.");
      scanf("%d", &n[i]);
      sum+=n[i];
    }
    printf("Sum is %d", sum);
}
```

Homework:

1) WAP to take input of 10 integer and calculate the average value input.

2) WAP to create multiplication table of 1 to 5.

Class work

3) WAP to access marks of 5 subject and calculate the percentage.

2) WAP to read 'n' number in array and display there sum and average.

3) WAP to find smallest and largest element in an array.

Home work - Imp lab sheet - 30

1) WAP to find the sum of the series

$$1 + x^2 + 3x^2 + 4x^2 + \dots nx^2$$

2) WAP to display the sum of the following series

$$Sum = x - x^2 + x^3 - x^4 + \dots$$

3) WAP to find the sum of the sequence

$$\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \frac{4}{4!} \dots \frac{n}{n!}$$

C.W 3) // Percentage of 5 subject

```
#include <stdio.h>
main(){
    int i, n[5];
    float sum=0, per;
  for(i=1; i<=5; i++){
  printf("Enter the marks:");
  scanf("%d", &n[i]);
  sum+=n[i];
  per=(sum/500)*100;
  }
  printf("The percentage of 5 subject is %.2f",
        per);
}
```

Library function : printf(), scanf(), getch().

Chapter
8

Function

- A function is a self contained block of statement that performs a per particular task.
- It is a logical unit composed of a number of statements grouped into a single unit.
-

Advantages:

i) Code reusability
ii) Logical clearity
iii) Non repeated programming
iv) Manageability
v) Easy to divide the work to many different program.

1) WAP to add two numbers using function

H.W Types of function with example / user defined function

i. Function with no return value and no arguments

- When a function has no arguments, it does not receive any data from the calling function.

- Functions that have no arguments and no return values. Such functions can either be used to display information or to perform any task on global variables.

Syntax:
```
void function_name() {
    // body of function
}
```

Example:

```
# include <stdio.h>
void sum ();
void main ()
{
  sum ();
  return 0;
}
void sum ()
{
  int a, b, s;
  printf("Enter two numbers:");
  scanf("%d %d", &a, &b);
  s = a + b;
  printf("sum = %d", s);
}
```

In the above program, function sum does not take any arguments and has no return values. It takes a and b as input from the user and prints them inside the void function.

ii) Function with arguments but no return type

- This type of function has arguments and receives the data from the calling function.
- Such functions are used to display or perform some operations on given arguments.

Syntax:

```
void function_name (argument_list)
{
  // body of function
}
```

Example:

```
#include <stdio.h>
void sum (int, int);
int main ()
{
 int a, b;
 printf("\n Enter two numbers :");
 scanf("%d%d", &a, &b);
 sum (a, b)
 return 0;
}
 void sum (int a, int b) {
   int s;
   s = a + b;
   printf("\n The sum is %d", s);
  }
```

iii) Function with Arguments and Return Type

- This type of function has arguments and receives the data from the calling function.
- These functions are used to perform specific operation on the given arguments and return their value to the user.

Syntax :

```
return_type function_name (argument_list)
{
  // body of function
}
```

Example :

```
# include <stdio.h>
int addition (int, int);
int main ()
{
   int a, b, s;
   printf("Enter two number : \t");
   scanf("%d%d", &a, &b);
   s = addition (a, b);
   printf("\n The sum is \t %d", s);
   return 0;
}
 int addition (int a, int b) {
    int s ;
    s = a + b;
    return s;
}
```

iv) Function with Return Value and no Arguments

- This function cannot pass data from calling function to called function.
- Such functions are used to perform specific operations and return their value.

Syntax:
```
return_type function_name(){
    //program
    return value;
}
```

Example:

```
#include <stdio.h>
int sum();
int main()
{
 int r;
 r = sum()
 printf("Sum = %d \n", r);
 return 0;
}
int sum(){
 int s, a, b;
 printf("Enter two numbers \n");
 scanf("%d%d", &a, &b);
 s = a+b;
 return s;
}
```

Questions for exam

1) What is programming language. Explain it's types.
2) What is program error. Explain with types.
3) Features of good program
4) Explain program development life cycle.
5) What are the different system design tool explain. [DFD, Discussion table, discussion tree, content diagram.
6) Software Process model. Explain it
7) Different between Top-down and bottom-up.
8) " " Cohesion and cupling
9) Features of C programming language.
10) What is data type. Explain different data type in detail.
11) Shor note:
    i) Token
    ii) Identifier
12) What is operator. Explain with type.
13) DO-while program.
14) ascending order, descending
15) Sorting
16) Prime number
17) No parameter return value.

Imp

1) // Factorial with argument with return value

```
#include <stdio.h>
main () {                    int fact (int)
  int n, r;
  printf ("Enter the number : ");
  scanf ("%d", &n);
    r = fact (n);
  printf ("Factorial = %d", r);
}
int fact (int a) {
    int i, f = 1;
    for (i = 1; i <= a; i++) {
        f *= i;
    }
    return f;
}
```

2) // Factorial with no argument return value

```
#include <stdio.h>
int fact ();
main () {
    int result = fact ();
    printf ("The factorial %d", result);
}
int fact () {
  int n, i, f = 1;
  printf ("Enter the number:");
  scanf ("%d", &n);
  for (i = 1; i <= n; i++) {
    f = f * i;
  }
  return f;
}
```

3) // Factorial with argument no return value

```
#include <stdio.h>
int fact (int b);
main () {
    int n;
    printf ("Enter the number ");
    scanf ("%d", &n);
    fact (n);
}
int fact (int b) {
  int i, f = 1;
  for (i = 1; i <= b; i++) {
    f = f * i;
  }
    printf ("The factorial of %d = %d", b, f);
}
```

4) // Factorial with no argument with no return value

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

```
int fact() {
  int i, f=1, n;
  printf("Enter the number :");
  scanf("%d", &n);
  for(i=1; i<=n; i++) {
  f*=i;
  }
  printf("Factorial of %d = %d \n", n, f);
  }
```

multiplication table of N numbers until

```
#include <stdio.h>
main() {
   int i, j, n, r;
   printf("Enter the Number :");
   scanf("%d", &n);
  for(i=1; i<=n; i++) {
   printf("~~An~~ Multiplication table of %d \n", i);
  for(j=1; j<=10; j++) {
   r=i*j;
   printf("%d * %d = %d \n", i, j, r);
  }
    printf("\n");
  }
  }
```

* Recursion (v.v.imp)

When a function is called from ~~inside of~~ same ~~it's~~ function or if it is called by itself

for e.g

```
main() {
  int a, b;
  add()
  ..
  add() {
  add();
  }
```

Recursion - factorial
- fibonic series

The condition has to be clearly explained with in the function otherwise the execution will continue indefinately.

1) WAP to find factorial of a number using recursion

```
#include <stdio.h>
int factorial (int);
main () {
  int i=5, result;
  result = factorial (i);
  printf ("The factorial is %d", result);
  }
int factorial (int n)
  {
  int fact;
  if (n==0)
    return 1;
  else
    fact = n * factorial (n-1);
  return fact;
  }
```

2) WAP to add two 2×2 matrix

```
#include<stdio.h>   => wrong xa book herni
main () {
    int a[2][2], b[2][2], c[2][2];
    c[0][0] = a[0][0] + b[0][0];
    c[0][1] = a[0][1] + b[0][1];
    c[1][0] = a[1][0] + b[1][0];
    c[1][1] = a[1][1] + b[1][1];
  }
```

* What is structure?

A structure is the collection of variables under one name.
It is a hetrogenous collection of related data which share the common name.

Syntax

```
struct stud {
    int id;
    float marks;
    char name[10];
  };
```

How to access the members of structure

The structure member can accessed by using dot (.) operator

For e.g

```
struct stud {
    int id;
    float marks;
    char name[10];
  };
main () {
  struct stud P1;
  printf ("Give data of P1");
```

```
scanf("%d", &P1.id);
scanf("%d", &P1.marks);
scanf("%d", P1.name);
}
```

Feature of file handling

i) Reusability
ii) Portability
iii) Efficient
iv) Storage capacity

Types of file
- i) Text file
- ii) Binary file

] Different between

↳ The binary file can be created only from within a program and their content can be only ~~write~~ be read by a program.

* What are the different file operations in C program:
  - i) Creating a new file - fOpen() with attributes "a", "a+"
  - ii) Opening a existing file, fOpen().
  - iii) Reading from file - fscanf() or fgets()
  - iv) Writing to a file - fprintf() or fputs()
  - v) Moving to a specific location in a file - fseek(), rewind()
  - vi) Closing a file - fclose().

```
#include <stdio.h>
#include <conio.h>
main(){
  FILE * fptr
  fptr = fopen("filename.txt", "r");
  if(fptr == Null){
    printf("File doesn't exist");
  }
}
```

read = while((c = fgetc(fp)) != EOF)

write = while((c = getchar()) != '\n') → lekhneka lagi 'c' maa c

fputc(c, fp)

File Handling

File

↳ The file allows to store information for permanently and to access the information whenever necessary.

↳ Basic file operations are i) opening a file
ii) Reading data from file
iii) Writing data to a file
iv) Closing a file.

File opening modes

i) 'r' mode

↳ This mode allows a file to open for reading only purpose.

Syntax:
file pointer = fopen ("file name", "r");

ii) "w" mode

This mode opens a file for writing only purpose

syntax:
file ptr = fopen("file name", "w");

If file doesn't exist. Then new file will be created.

iii) "a" mode:

This mode opens an existing for appending ~~file~~ purpose, appending means add data at the last.

~~append~~ Syntax:

If file doesnot exist new file is created

iv) "r+" mode: (read + write)

This mode is used to both read & write
Syntax:
file ptr = fopen ("student.txt", "r+");
It show error if file doesn't exist.

v) "w+": (write + read)

Syntax:
file ptr = fopen ("stud.txt", "w+");

If file doesn't exist it will create the file.

Matrix multiplication

```
#include <stdio.h>
int main ();
{
 int a[2][2], b[2][2], c[2][2], i,j, k, s=0;
 fo printf("Enter the matrices");
 for (i=0; i<2; i++){
  for (j=0; j<2; j++)
  {
  Scanf("%d%d", &a[i][j], &b[i][j]);
  }}
for (i=0; i<2; i++){
  for (j=0; j<2; j++{
  s=0;
  for (k=0; k<2; k++){
  s=s+a[i][k] * b[k][j];
  c[i][j]=s;
  }}
for(i=0; i<2; i++){
for (j=0; j<2; j++){
printf("%d", c[i][j]);
}
 printf("\n");
return 0;
 }
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 12 \\ 43 & 50 \end{bmatrix}.$$

$$\begin{bmatrix} 1\times5+2\times7 & 1\times6+1\times8 \\ 3\times5+4\times7 & 3\times6+4\times8 \end{bmatrix}$$

C - Programming
Assignment

1) WAP that computers the sum of digits of given integer number.

```
#include <stdio.h>
main (){
   int temp, sum = 0, num;
   printf("Enter a integer number :");
   scanf("%d", &num);
   while (num != 0){
      temp = num % 10;
      sum += temp;
      num = num / 10;
}
   printf("The sum of digits of given
          integer sum);
   return 0;
}
```

2) What are the three types of input/output function which supports in C-programming? Explain with Example.

Ans The function which are used to read data from keyboard are called inputs functions. Some standard input functions includs: scanf(), getchar(), getche(); getch(); gets(), etc.

Similarly the functions which are used to display result on the screen are called standard output functions. Some of them are: printf(), put char(), et

Types:

i) Formatted I/O
The functions which allow input data to formate according to user's requirements are known as formatted I/O function.

a) Formatted Input:
Formated input functions are used to read data from a standard input device. scanf() is used for formatted input operation. The general form of scanf().

scanf(*control string, arg1, arg2);

b) Formatted output function
They are used to display or store data in a particular specified formate. The printf() is an example of formatted output function.

ii) Unformatted I/O
Unformatted I/O functions donot allow to read or display data in desired formate. These type of library functions basically deal with a single character or a string of characters. The function getchar(), putchar(), gets(), puts(), getch(), putch() are some examples of unformatted functions.

2) Discuss logical operator along with the truth table.

ans In addition to the relational operators, char the following three logical operators.

&& meaning logical AND
|| meaning logical OR
! meaning logical NOT

The logical operators && and || are used where we want to test more than one condition and make decision.

Example:

a>b && x==10

An expresion of this kind, which combines two or more relational expressions or a compound relational expression.

5) What is looping statement? Discuss different looping statement with suitable example of each.

Ans Looping statement is the process of executing a group of statement more than one time as long as some condition remains true loops are used when we want to execute a part of program or block of statement several time.

Types:

i) For loop
ii) While loop
iii) do while loop.

i) While loop

The while statement is typically used in situations where it is not known in advance how many iterations are require. A while loop is the most basic type of loop. It will run as long as the condition is nonzero (true).

Syntax:

```
while (condition) {
    statement body;
}
```

ii) Do-while loop

Do while loop executes a body, first without checking any conditions and then checks a test condition to determine the body of loop is to be executed. For next time or not. In this loop, the statement body is always executed at least one.

Syntax:

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

3. The for loop

The for loop is useful to execute a statement for a number of times when the number of repetations is known in advance, the use of this loop will be more efficient. Thus, this loop is also known as a definite loop.

Syntax:

```
for (initialization; condition; increment/decrement)
{
    statement body;
}
```

6) Discuss different type of if statement with example of each.

Ans The if statement is a two-way decision statement & is used together with an expression i.e test condition. The if statements evaluates the test expression first and then, if the value of the expression is true, it executes the statement within is block & continue from the first statement outside the if block.

General Syntax:

```
If(test - expression){
statement - block;
}
```

7) Differentiate if-statement with switch statement.

If statement	Switch-case statement
i) It is used for evaluating relational or logical condition.	It is used for selecting among multiple values of a single variable usually integer or character
ii) It can evaluate complex expression (e.g if $x>5$ && $y<10$)	It can only evaluate simple equality condition (e.g: switch)
iii) It supports multiple datatypes (integers, floating point, string, etc)	It is typically support only integer, character & enumerator data type.
iv) It may become harder to read when there are many conditions.	It is easier to read when dealing with a large number of discrete values.
v) In this, once a condition is met, subsequent blocks are ignored.	fall through occur unless a break is used after each case.

7) Differentiate between break & exist statement with example.

Break	Exit
i) Terminates the current loop or switch statement and transfer control to the next statement following the loop or switch.	Terminates the entire programs immediately, regardless of where it is called & returns control to the operating system.
ii) It affects only the loop or switch in which it is called.	It affects the entire program ending all operations and closing the applications.
iii) Used to control the flow within loops ~~if~~ or switch - case statements.	Used when you want to completely stop the program.
iv) Doesnot return any value since it simply transfers control within the program flow.	Can pass an exits status or return code to the OS.
v) After execution, the program continues to run	After execution, the program will terminate.

8) Differentiate between while and do-while loop

The while loop	The do-while loop
i) The while loop is entry control loop.	This loop is exist control loop.
ii) This loop will execute only if the test condition is true.	The loop will execute for first time without checking test condition.
iii) The body of the loop may not be executed at all if the condition is not satisfied at the very first attempt.	The body of the loop is always executed at least once.
iv) Syntax: while (condition) statement body;	Syntax: do { statement body; } while (condition);

v) Example.
#include <stdio.h>
main (){
int x=1
while (x<=10){
printf ("%d \t", x);
x++;}
}
	vi) Example:
	#include <stdio.h>
	~~void~~ main ()
	{
	int x=1;
	do {
	printf ("%d \t", x);
	x++;}
	while (x<=10);
	}

9) Define array? What are the benefit of array? Write a program to add two matrices using array.

Ans An array is a collection of elements of the same data type placed in contiguous memory location and can be accessed individually using index to a unique give name. They are group of realted data items that share a common name.

Benefits:

i) Array can store a large number of value with single name.
ii) Array are used to process many value easily and quickly.
iii) The values store in an array can be sorted easily.
iv) The search process can be applied on array easily.

// Adding ~~to~~ two matrices

```
#include <stdio.h>
main () {
   int a[3][3], b[3][3] sum[3][3], i, j;
   printf("Enter the element of first matrix:");
   for (i=0; i<3; i++){
   for (j=0; j<3; j++){
   {
    scanf ("%d \t", &a[i][j]);
   }
   printf("\n");
   }
```

```
printf("Enter the elements of 2nd matrix:");
for(i=0; i<3; i++){
 for(j=0; j<3; j++){
 scanf("%d \t",&b[i][j]);
 }
 printf("\n");
}
// for sum
 for(i=0; i<3; i++){
 for(j=0; j<3; j++){
 sum[i][j] = a[i][j] + b[i][j];
 }}
printf("The sum of the matrices is:\n");
for(i=0; i<3; i++){
for(j=0; j<3; j++){
 printf("\t %d", sum[i][j]);
 }
printf("\n");
}.
```

(16)

11) WAP to input any 10 number then find out greatest & smallest number.

```
#include<stdio.h>
main(){
int i, n a[10], s, l
print for(i=0; i<10; i++){
printf("Enter the %d the element number:", i+1);
scanf("%d", &a[i]);
```

10, 13, 14, 16, 17, 1

```
small s = a[0];
      l = a[0];
for(i=1; i<n; i++){
 if(small > a[i])
  small s = a[i];
  if(larg < a[i])
  larg l = a[i];
 }
 printf("\n The smallest element = %d and the
largest element = %d", s, l);
}
```

15) What is recursion? WAP to find the factorial of given number using recursion.

Recursion is the process in which the program repeats a certain section of code in a similar way.
The definition that defines an object in terms of ~~similer~~ simpler case of itself is called a recursive definition.

// finding factorial

```
#include<stdio.h>
main(){ long fact(int);
 int a;
 printf("\n Enter a number:");
 scanf("%d", &a);
 printf("\n the factorial of the %d number is
 %ld", a, fact(a));
```

```
}
long fact (int n)
{
  if (n == 0)
  return 1;
   else
  return (n * fact (n-1));
}
```

16. What is structure? How is it different from array.

Ans Structure is a heterogenous collection of related data which share the common name.

Array	Structure
i) Array is a collection of homogeneous data.	It is a collection of heterogenous data.
ii) Array data are access using index.	Structure elements are access using operator.
iii) Array allocates static memory.	Structure allocates dynamic memory.
iv) Array element access takes less time than structure.	Structure elements takes more time then array.

19. Why do we need data files? What are different file opening mode? WAP that read data from a file "input.txt" and write to "output.txt" file.

Ans Many application require that information be written to or read from an auxillary storage device. Such information is stored on the storage device in the form of data file. Thus, data files allow us to store information permanently and to access later on and alter that information whenever necessary.

// Program to copy file

```
#include <stdio.h>
main () {
    FILE *sfp, *dfp; char c;
    char sfp = fopen ("input.txt", "r");
     if (sfp == Null) {
     printf ("File cannot open");
      exit (.);
    }
    dfp = ("output.txt", "w");
     if (dfp == Null) {
     printf ("FILE cannot created");
      exit ();
    }
     while ((c = fgetc (sfp)) != EOF)
     fputc (c, dfp);
     printf ("\n Copied ---");
     fclose (dfp);
     fclose (sfp);
}
```

9) WAP to read and print data store in a file input.txt.

```
#include<stdio.h>
main(){
  FILE *fp;
  char s[100];
  fp=fopen("C:\\input.txt","r");
  if(fp==NULL){
  printf("\n cannot open file");
   exit(); }
else{
    printf("\n file is opened");
  }
fgets(s,79,fp);
printf("\n Text from file is : %s",s);
 fclose(fp);
}
```

10) WAP to check whether the diagonal elements of a (4*4) matrix all zero.

```
#include<stdio.h>
main(){
 int a[4][4],i,j,f=1;
 printf("Enter the elements of matrix:\n");
 for(i=0;i<4;i++){
 for(j=0;j<4;j++){
  scanf("%d",&a[i][j]);
 }
   printf("\n");
  }
 for(i=0;i<4;i++){
 if(a[i][i]!=0){
   f=0;
   break;}}
 if(f){
    printf("All diagonal elements are 0.\n");
 }else{
    printf("Not all diagonal elements are 0.\n");
  }
}
```

Different between Cohesion and Coupling

Aspect	Cohesion	Coupling
Define	Degree to which elements of a module belong together.	Degree of interdependence between modules.
Focus (i)	Internal functionality within a module.	Interaction between different modules.
Goal (ii)	High cohesion is desirable	Low coupling is desirable
(iii)	Easier to maintain and modify	Harder to maintain if highly coupled
(v)	Module performs a single task well.	Modules are highly dependent on each other.
	Example: A function handling one task.	Two classes tightly interacting.

The process of allocating and freeing memory at the run time is known as Dynamic Memory Allocation. This reserve the memory required by the program and returns this resource to the system once the use of reserved space utilized. ~~The process of allocation~~

* library function for memory management

i) Malloc ( )
Syntax: p = (data_type*) malloc (size_of_block);

ii) free ( )
Syntax: void free (void *p);

iii) Calloc ( )
e.g x = (int*) calloc (5, 100 * sizeof(int));

iv) Realloc ( )
⇒ To add more space and decrease space.
Syn: ptr = realloc (ptr, newsize);

Graph:

line (int x1, int y1, int x2, int y2);

circle (int x, int y, int r);
(x,y) are center of circle.

Note:
Full screen: maxx = getmaxx();
maxy = getmaxy();
For center;
x = maxx/2;
y = maxy/2;
circle (x, y, 50);

Drawing ellipses

ellipse (int x, int y, int starAngle, intendAngle, int xRadius, int yRadius);

(x, y) → center point of ellipse
starAngle → starting angle in degree
EndAngle → Ending angle in degree
xRadius → radius along x axis (—),
yRadius → radius " y axis (|).

Circular arc

~~arc = (~~ arc (int x, int y, int starAngle, intendAngle, int radius);

Rectangle:
rectangle (int x1, int y1, int x2, int y2);

(x1, y1) → left top corner point of rectangle.
(x2, y2) → right bottom " " " "

Drawing & filling polygon

drawpoly (int number of points, int points[ ]);
fillpoly (int " " " , int points[ ]);

WAP to draw hexagon

```
#include<graphics.h>
main(){
    int gd, gm;
    int gd = DETECT;   // 14 ota number kuna para
    int poly[ ] = {10, 5, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};
```

initgraph(&gd, &gm, "C:\\tc\\bgi");
drawpoly(7, poly);
Fillpoly(7, poly);
closegraph();
}

Display text in graphics mode.

i) outtext(): Display the string at current position
outtext(string text);
exmp: outtext("Hello BCA");

ii) outtextxy(): display string at point (xy).
syn: outtextxy(int x, int y, string text);
eg: outtextxy(50, 50, "This is test");

iii) settextstyle(): change font, size and direction of character.

syn: settextstyle(int font, int direction, int size);
eg: settextstyle(GOTHIC_FONT, HORIZ_DIR, 2);

for color;

~~setcolor(color name);~~

setcolor(): change current drawing/foreground color)

setcolor(int color);
e.g setcolor(RED);
setbkcolor(): change background color()
setbkcolor(int color);
e.g setbkcolor(BLUE);

Drawing a sector

syn: pieslice(int x, int y, int startAngle, int endAngle, int r);

* WAP to store the odd number in odd file and even number in the even file and read the number of odd file.

```
#include<stdio.h>
main(){
  FILE *of, *ef;
  int i, num, n;
  printf("Enter the numbers of elements you want
        to store");
  Scanf("%d", &num);
  of = fopen("Odd.txt", "a");
  ef = fopen("even.txt", "a");
  for(i=0; i<num; i++){
  printf("Enter the %d number :", i+1);
  scanf("%d", &n);
  if(n%2 == 0){
  fwrite(&n, sizeof(int), 1, ef);
  }
  else{
   fwrite(&n, sizeof(int), 1, of);
    }
  fclose(of);
  fclose(ef);
  //reading number
  of=fopen("odd.txt", "r");
  while(fread(&n, sizeof(int), 1, of)==1)
  printf("%d \n" n);
  fclose(of);}
```

* WAP to draw two concentric circles with center (50,50) & radii 75 and 125.

```
#include <stdio.h>
#include <graphics.h>
main (){
    int gd = DETECT, gm;
    initgraph (&gd, &gmode, "C:\\tc\\bgi");
    circle(50, 50, 75);
    circle(50, 50, 125);
    outtextxy(x, y, "");
    closegraph();
}
```

2078

A structure is a convienent way of grouping several pieces of related information together.

A structure is a collection of variables referenced under one name, providing a convenient means of keeping related information together

Union is heterogenous collection of data elements related to same entity under the name of same variable.

// Starting Program

```
#include<stdio.h>
#include <conio.h>
Struct student{
    int r, m;
    char a[40];
    char n[40]; }, s[35];
main (){
    int i;
for (i=0; i<35; i++){
 printf("Roll Number : ");
 scanf("%d", &s[i].r);
 printf("Give name : ");
 scanf("%s", s[i].n);
 printf("Give address : ");
 scanf("%s", s[i].a);
 printf("Enter the marks:");
 scanf("%d", &s[i].m);
 printf("\n");
}
// Now searching data
printf("The students whose marks greater than 250 : ");
for (i=0; i<35; i++){
 if (s[i].m > 250){
  printf("Roll : %d, Name : %s, Address : %s, marks : %d \n", s[i].r, s[i].n, s[i].a, s[i].m);
 }
}
}
```

* WAP to find the sum of all odd number from 1 to 100.

```
#include<stdio.h>
main (){
    int i, sum = 0;
for (i=0; i<100; i++){
```

```
if (i%2 == 1){
    sum = sum + i;
}
printf ("sum is %d", sum);
}
```

* WAP to find the sum of two matrix order of m×n.

```
#include<stdio.h>
main ()
{
  int a[5][5], b[5][5], c[5][5], i, j, m, n;

printf("Enter the order of matrix (less than 5*5)";\n);
scanf("%d", &m);
scanf("%d", &n);
for // Now entering the variables

for(i=0; i<m; i++){
for(j=0; j<n; j++){
scanf("%d", &a[i][j]);
scanf("%d", &b[i][j]);
}}
// Now adding

for(i=0; i<m; i++){
for(j=0; j<n; j++){
c[i][j] = a[i][j] + b[i][j]; }}
printf("Sum = %d", c[i][j]);
}
```

strcpy ⇒ copy,
strcat ⇒ combin
strcmp ⇒ compar
strupr ⇒ upperletter
strlwr ⇒ lowercase
strlen ⇒ measure length.

ascending in name;

```
for(i=0; i<5; i++){
for(j=i+1; j<5; j++){
if (strcmp(s[i].name, s[j].name)>0){
    temp = s[i];
    s[i] = s[j];
    s[j] = temp;
}}}
```

Diff struct & union ⇒ page 282

The function which allow input or output data to format according to user's requirement are known as Formatted I/O functions. The input function scanf() and output function printf() are formatted I/O function.

Formatted input function can be use to get input from standard I/O device.

formatted output function can be use to display and store data in a particular speed format.

Unformatted I/O functions do not allow to read and display in desired format. These types of library functions basically deal with a single character or string of character. char read, string read

Unformatted Input function = getchar(), getch(), getche() for char
" output = putchar(), putch(), puts() → string display
→ character display

Step used in programming Development.

The program development life cycle is a set of steps ~~are~~ ~~called~~ or phases that are used to develop a program in any programming language.
→ i) Problem analysis, ii) Algorithm Development, iii) Drawing Flowchart, iv) Writting source code (coding), v) compilation & execution, vi) Debugging & Testing vii) Support and maintanance, viii) Documentation.

Important note

i) While using fread() and ~~fwrite~~ fwrite() function use binary mode such as ab → append, wb → write, rb → read

ac :- br

* Concentric circle in graph
write your name in rectangle center

* structure — always long question
i) IO input array
ii) Record the data in file
iii) Read the data from file
iv) Using pointer

* File handling
⇒ Create file and write "Hello"
⇒ In file there is "My name is Bikram Oyawali" in a file now count ~~the~~ how many are letter.

* Palindrome number or not
* Amstrong number
* Fibonic Series

* Ascending order
* Descending order
* Bubble shoot
* add matrix
* Multiply matrix
* Diagonal same or not
* Matrix element are same or not.

plang
* what is flowchart, algorithm, model, iteration, recursion
→ factorial, Fibonic Series, in recursion.