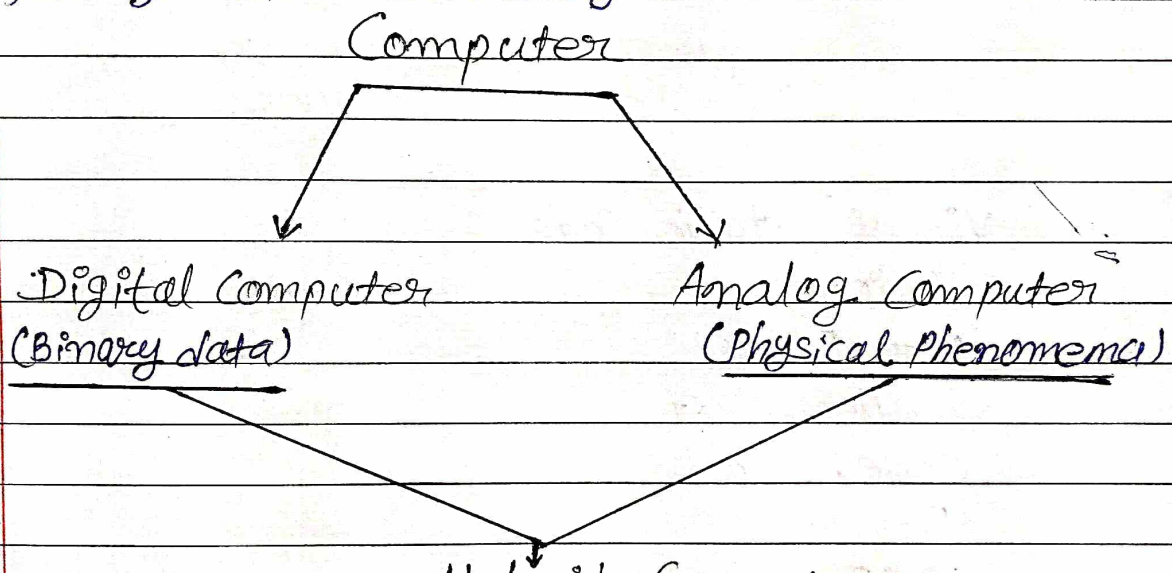
Programming for Problem Solving
ESC-104

Page No. 1
Date 10 / 2022

Introduction to Programming

* What is Computer ?

Ans ⇒ Computer is a electric machine which is Compute data.
When we give input data to computer then computer process this data and give output information. It can solove highly complicated problems quickely and accurately.



* What is Digital Computer ?

Ans ⇒ Digital computer is an electric machine which accept data in input and process it and produce information in output.

* What is Software ?

Ans ⇒ Software is a set of instruction written in specific Computer understandable form and and preform specific task.
Software → programming → C, C++, Java, Pythem etc.
Computer understand only (0,1)

Page No.
Date / 20

* What is programming language?

Ans ⇒ Language in which we can design program Software for computer is called as programming language.

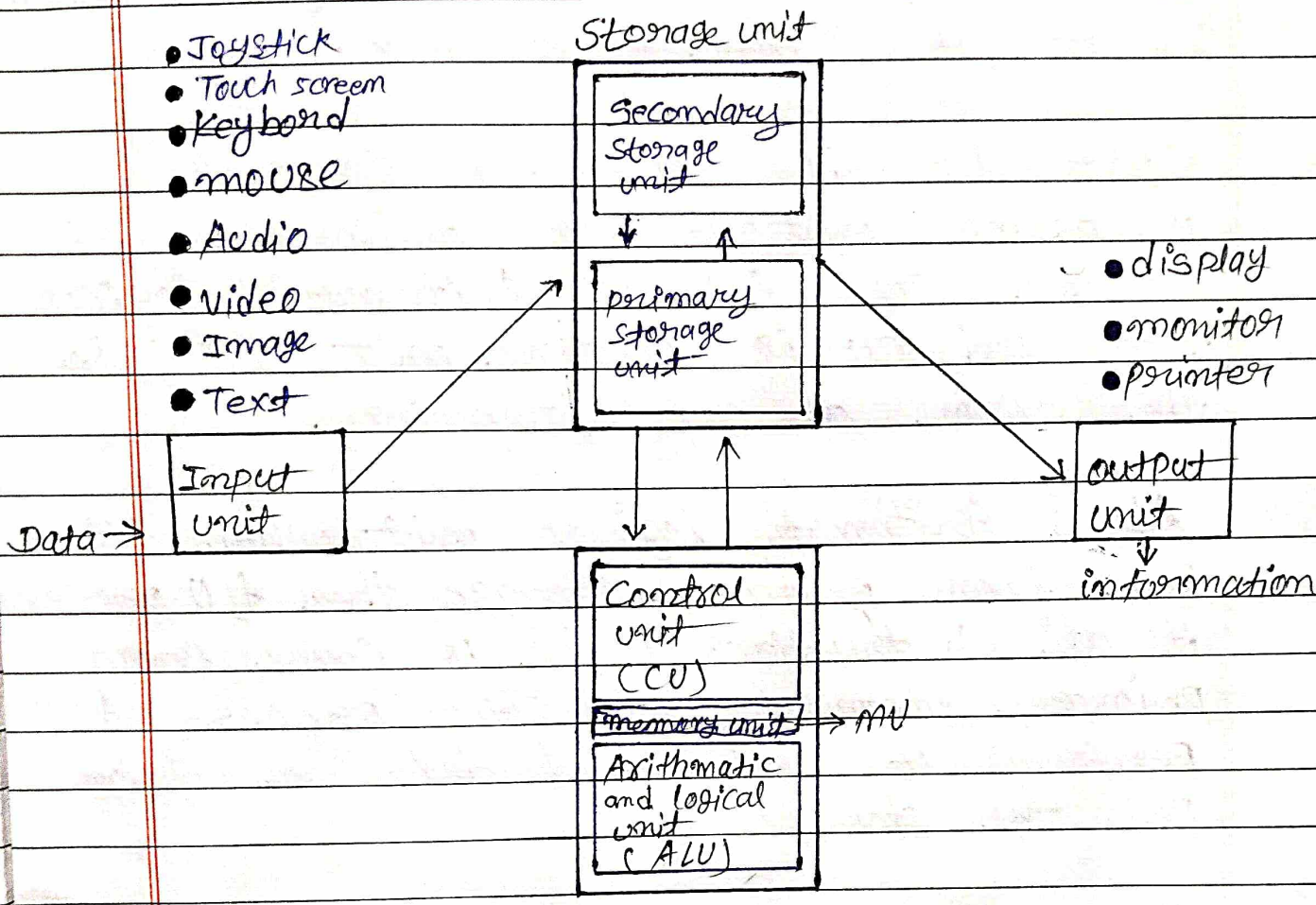
* Translator → (i) Assembler
(ii) complier
(iii) Interpreter

Top IDE for C language

(i) Visual studio code
(ii) Eclips
(iii) Net Beans
(iv) Sublime Text
(v) Code: Blocks
(vi) Dev c++
(vii) ~~T~~ Turbo C++
and Many more

Page No.
Date / 20

* Introduction to Components of a computer system and block diagram of computer.



(i) Input :- Through input unit we give instruction for computer that what i want from computer. And that instruction will be text, Audio, video, Image and i can give instruction for computer through keyboard, mouse.

(ii) (Storage unit) → Primary memory :- Basically storage unit is the combination of primary and secondary memory. Primary memory also known as main memory. RAM can be primary memory. Primary memory holds data as long as electric flows through it.

Page No.
Date / 20

(iii) Secondary memory :- In this memory our data save in this secondary memory for life time and for future and we can say it is second input unit.

(iv) CPU :- CPU is known as Central processing unit. It is combination of ALU and CU. ALU is Arithmatic logical unit and CU is Control unit. CPU is the brain of any computer.

(v) ALU :- Arithmatic logical unit when data come from primary storage then ALU process it all instructions which is come from primary memory and after process ALU perform it and send data in again primary storage.

(vi) CU :- Control unit is like supervisor it is supervise all data and seeing also hanging and bugs problem and. It ~~is~~ Controls it.

(vii) Output unit :- Output unit, when data after process by CPU comes in primary memory then output unit display it and show it. It is known as monitor.

* primary memory → ROM, RAM, Cash memory

Page No.
Date / 20

* What is ROM?

ans ⇒ ROM is Read Only memory is a type of non-volatile memory used in Computer and other electric device. Data stored in ROM ~~Cannot~~ Cannot be change and modified after the manufacture of the memory device.

** What is RAM?

ans ⇒ RAM → random-access memory, RAM is volatile. That mean data is retained in RAM as long as the Computer is on, but it is lost when the Computer is turned off. RAM is like a Plateform these app, video, audio, song, pdf perform. when we open any app or video than data come to the RAM from HDD or SDD and play in RAM. RAM like memory but not permanent.

returntype → header files

```
#include <conio.h>
int main()
{
    printf("Hello World");
    return 0;
}
```

function

indentation

Contents closed in curly braces is body of the function.

Write the program to add three number and apply addition on average

Page No.
Date / 20

*
```
#include <stdio.h>
int main()
{
  int a = 45;
  int b = 66;
  in c = a+b;
printf("the sum is %d", c);
  return 0;
  }
```
after run → The sum is (111)

*
```
# include <stdio.h>
  int main()
  {
  int a = 5;
  int b = 7;
  int c = 8;
  int d = a+b+c;
  printf("the sum is %d", d);
   return 0;
   }
```
after run → the sum is 20

*
```
# include <stdio.h>
  int x = 5;
  Int y = 7;
  int z = 8;
  int main()
{int a = x;
  int b = y;
  int c = z;
   int d = a+b+c
printf("the sum is %d", d); return 0;
}
```

Page No.
Date / 20

after run → the sum is 20

*
```
# include <stdio.h>
int x = 5 ;
int y = 7 ;
int z = 8 ;
int p = 876 ;
int q = 790 ;
int r = 3576 ;
int main ()
{ int a = x + p ;
  int b = y + q ;
  int c = z + r ;
  int d = p + r ;
  int E = a + b + c + d ;
printf ("the sum is %d", d) ;
  return 0 ;
  }
```

after run result ⇒ the sum is -----

* Operating system :- Operating system is a system software which interact with hardware it provids a inteface between user and machine. Acts as a manger of the Computer system Operating system is like soul of Computer.

Page No.
Date / 20

* Advantage of Computer.

(i) Speed ÷ It can calculate millions of data and information ~~what~~ within a fraction of second.

(ii) Storage ÷ It can store large amount of data in storage.

(iii) Accuracy ÷ It can perform any task at very high speed without any mistake and gives accurate data and information.

(iv) Reliability ÷ The information stored in computer is available after years in same form it work 24x7 without any problem and mistake.

* Automation ÷ It perform any complicated task by a single click whereever you want.

* Multitasking ÷ It can perform more than one tasks at a time.

* Disadvantage of Computer ÷

(i) Unemployment
(ii) virus attack
(iii) Missing Physical interaction
(iv) eye problem
(v) Health issue
(vi) electricty loss
(vii) Expensive
(viii) Unable to correct mistake

Page No.
Date / 20

* What is Hardware ?

ans ⇒ Hardware is a physical part of any Computer. we can see and touch the hardware and its Components.

Ex ⇒ CPU, Keyboard, mouse, display, printer.

* What is Software ?

ans ⇒ Software is a set of instructions written in specific Computer understandble form and perform a specific task and along with this a set of program is called Software.

* Types of Software ?

ans ⇒ (i) System software
(ii) Application software

(i) * System Software :- System software is software which intract with hardware it provides interface and plateform between user and machine, system software is soul of Computer. System software is operating software.

(ii) Application Software :- An application software is a computer program to perform a specific personal, educational, business, entertenment, sports function. Ex ⇒ Whatsapp, facebook, Hotstar.

* Categories of System Software

(i) Operating System :- Operating System Control hardware as well as interacts with users and provids interface and plateform it is a bridge between user and Computer as well as soul.

Page No.
Date / 20

* System support software :- It makes working of hardware more efficiently.

* System development software :- It provides programing development environment to programmers.
Ex ⇒ Editor, pre-processor, Compiler.

* General purpose software :- It is used widely by many people for some task. It is designed on vast Concept so many people can use it.

* Special purpose sofware :- It is used by limited people for some specific task it is designed as per users requirement.

* Types of Computer language.
ary ⇒ (i) Low level language
(ii) Assembly language
(iii) Higher level langage

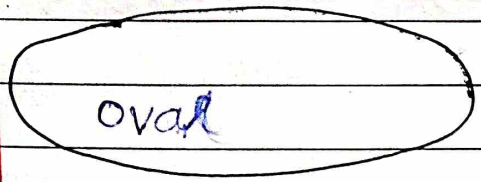
(i) Low lebel language or Machine language :-

It is language of 0 and 1 Computer directly understand this languag. It is non portable and it is special for machine it close to hardware it's work quickly.

(ii) Higher level language :- It is machine independent language and close to programmers. It is natural language like Ealphabet, number, symbal and it is portable.

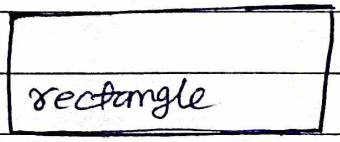
(iii) Assembly language :- Assembly language is little easier than machine language. In machine language if I have to add two number or subract two number or divide two number than we have to write many 0, 1 for (+, ×, −, ÷) but in assembly language we use mnemonics for (+) we use in place of add (−) we use in place of sub and assembly convert this mnemonics in to machine language. is called Assembly language.

* flowchart



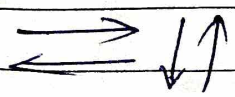
→ Start/End

→ input/output

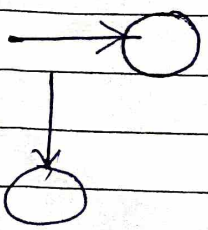


→ process (c = a + b)

→ Decision (if, else, switch)



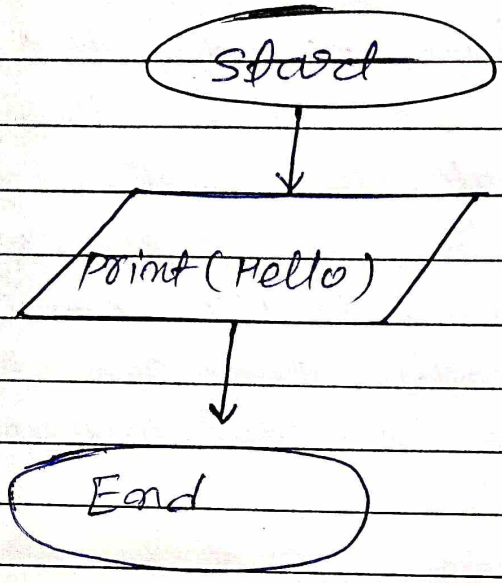
→ Dataflow



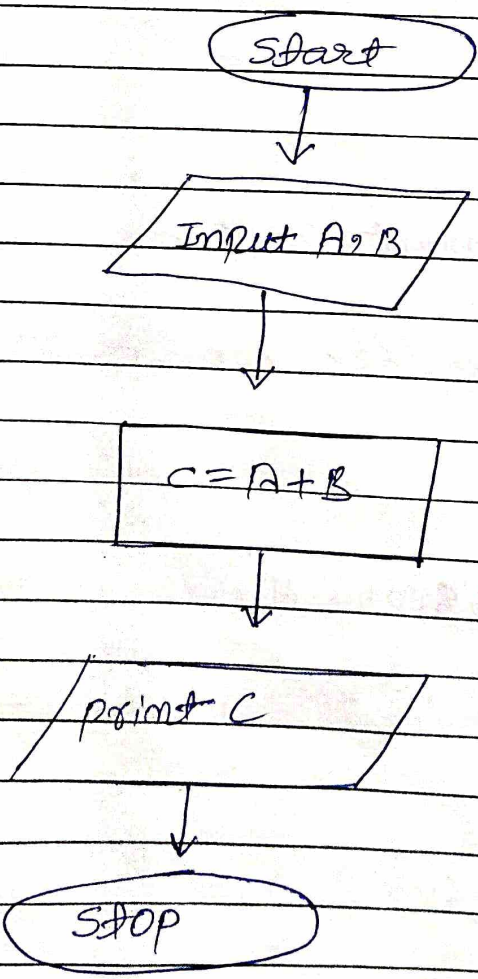
→ Connector

Page No.
Date / 20

* print Hello by using flowchart

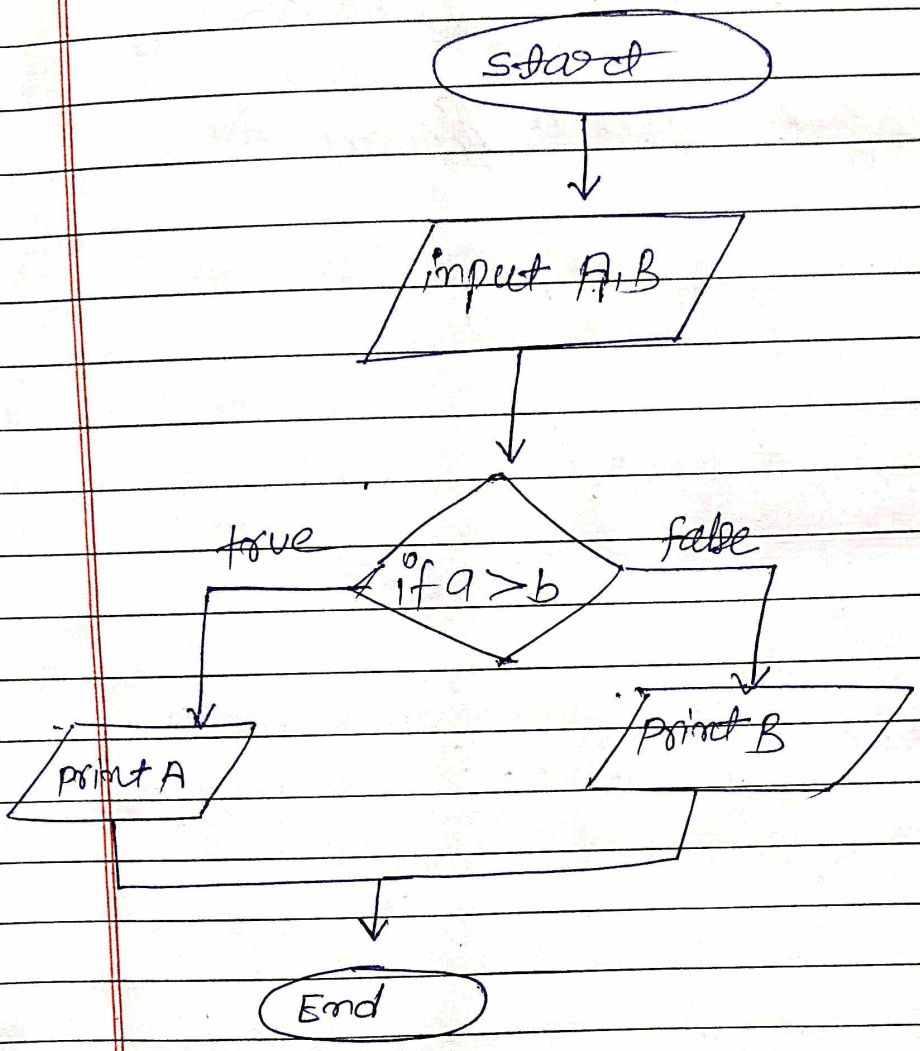


* add two numbers take data by user from flowchart

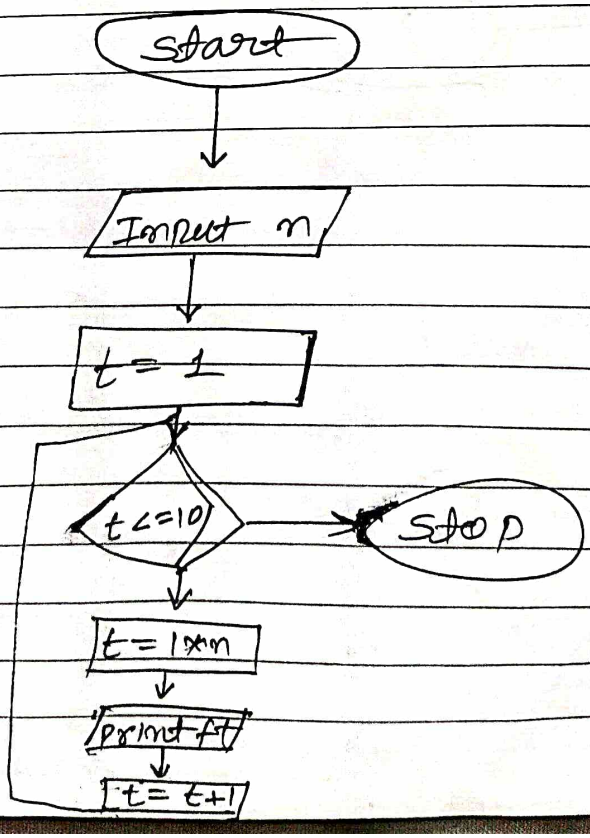


Page No.
Date / 20

* find greater number by user in flowchart

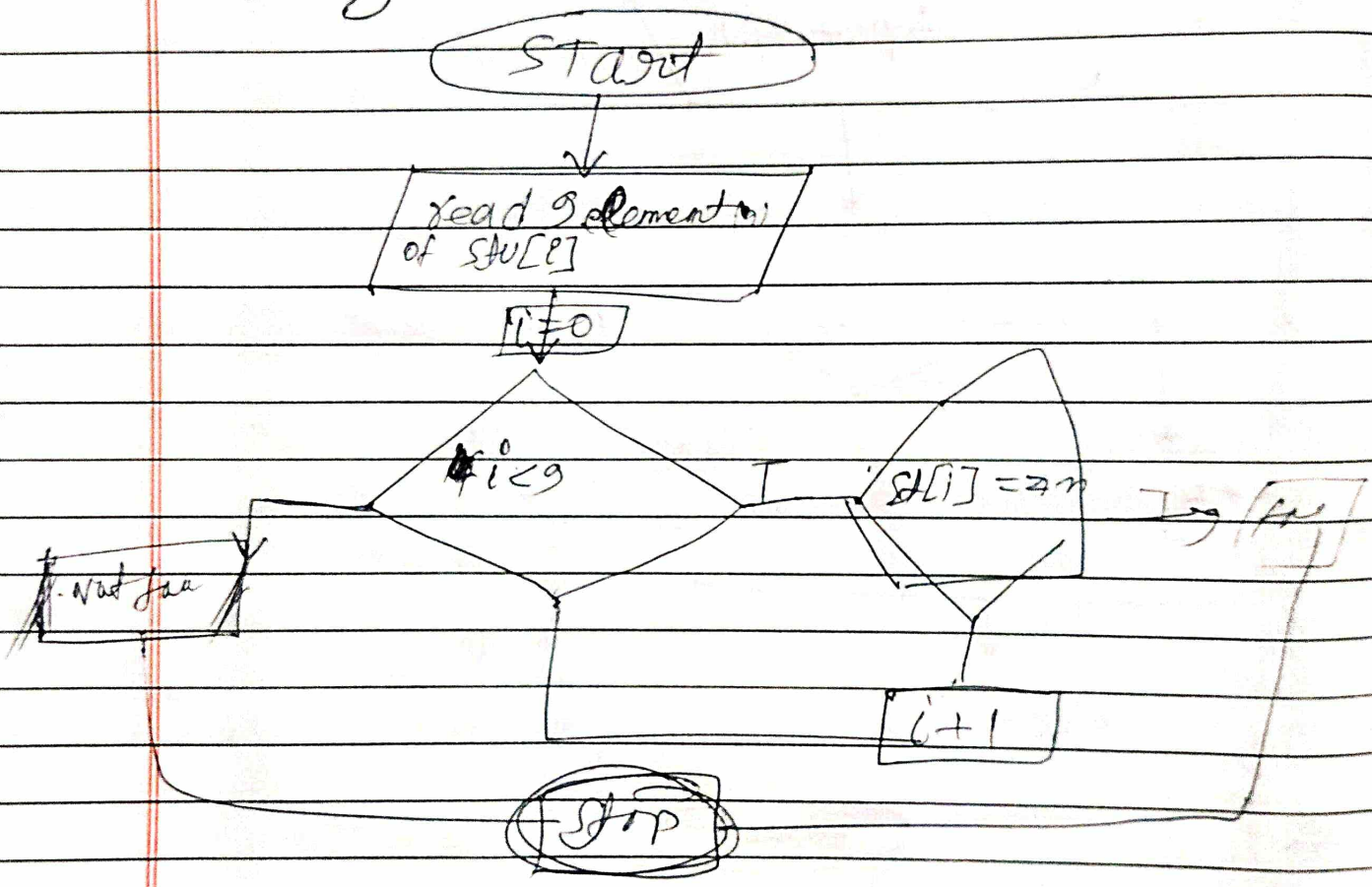


*



WAP to take input in array and find number in the array.

3rd largest num from the array



MST - 2

Page No.
Date / 20

* Searching Algorithm -

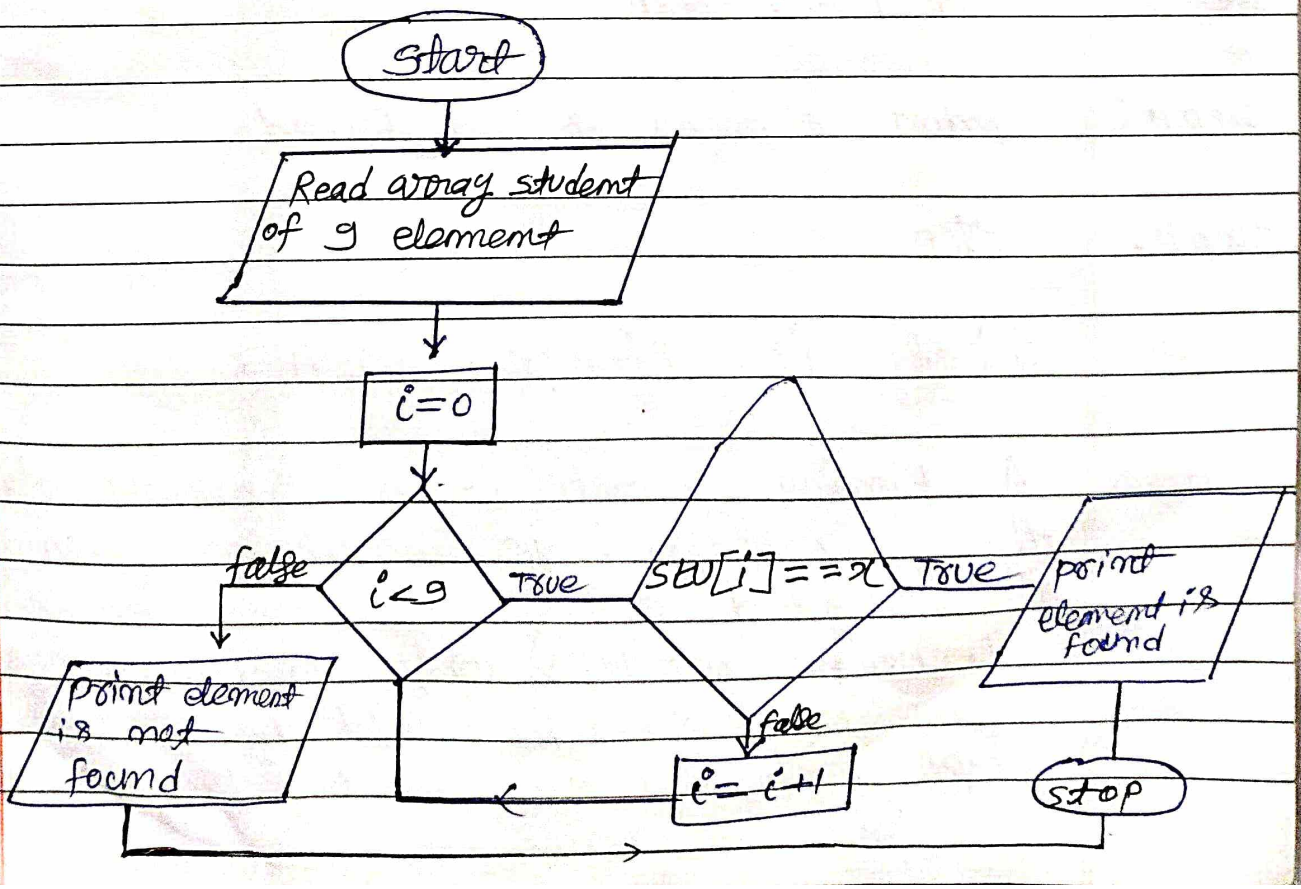
ans⇒ A search algorithm is any algorithm which slove the search problem.

* Types of searching algorithm
(i) Linear search
(ii) Binary search

(i) Linear search & (ii) Binary search

(i) Linear search : Linear search, also known as sequential search, it is a process that checks every element in the list sequentially until the desired element is found.

flowchart of linear search



Page No.
Date / 20

Algorithm of linear search

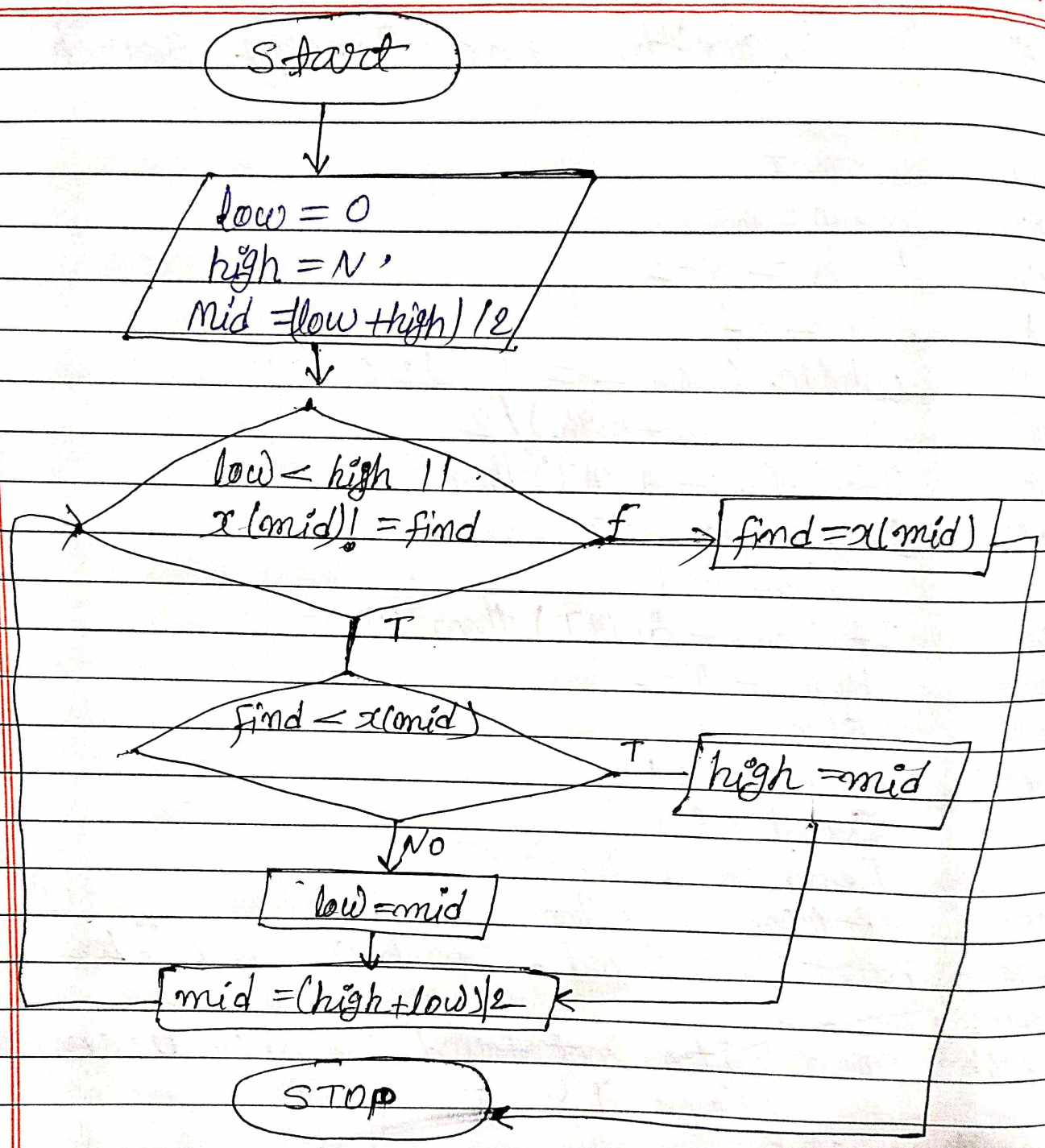
Step:1 → Start
Step:2 → input N
step:3 → for I ? 0 to N-1 Read A[I]
step:4 → [redacted] Loc = -1
step:5 → for I = 0 to N-1 do
Step:6 → If (ele = A[I]) Then
Step 7: Loc = I
Step8: Goto step 9
[end if]
[end for]
step9: If (Loc >= 0) then
Step 10: print "ele found in location", Loc
Step 11: else
Step 12: print "ele not found"
Step 13: stop

* Binary Search : Binary Search also known as half-interval Search, logarithmic Search or binary ~~chop~~ Sort is a search algorithm that finds the position of a target value within a sorted array. Binary Search compares the target value to the middle element of the array.

Page No.
Date / 20

* Algorithm for Binary Search

Step1: Start
Step2: Low = 0
step3: High = N-1
Step 4: Loc = -1
step5: while (Low <= High) DO
Step6: M = (low+High)/2
step7: If (ele = A[M]) then
step8: Loc = M goto step12
[end if]
step9: If (ele < A[M]) then 10
step10: High = M-1
step11: Else
step12: Low = M+1
[end of if]
[end of while loop]
step13: if (Loc >= 0) Then
step14: print "ele found, search successful"; Loc
step15: else
step16: print "Item not found, search unsuccessful"
[end if]
step17: stop


Start
low = 0
high = N
mid = (low + high) / 2
low < high ||
x(mid) != find
F
find = x(mid)
T
find < x(mid)
T
high = mid
No
low = mid
mid = (high + low) / 2
STOP

Page No.
Date / 20

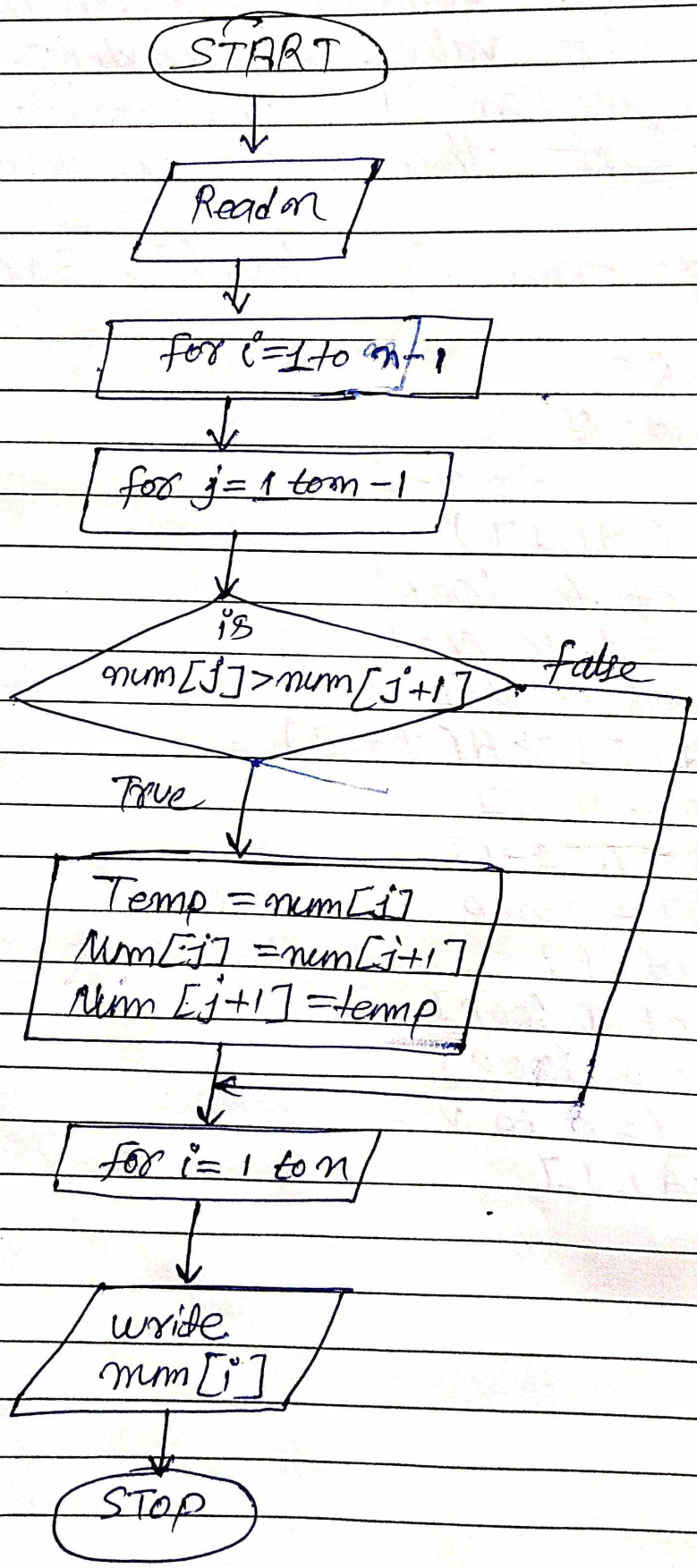
* Bubble Sort :- Bubble sort is a type of sorting element you can use to arrange a set of values in ascending order. If you want, you can also implement bubble sort to sort the values in descending order.

Algorithm for bubble Sort

Step 1: START
Step 2: Input N
Step 3: for I = 0 to N-1
Step 4: Read (A[I])
(end of for loop]
Step 5: for I = 1 to N-1
Step 6: for J = 0 to N-I-1
Step 7: If (A[I] > A[J+1])
Temp = A[I]
A[J] = A[J+1]
A[J+1] = Temp
[end of if]
[end of J loop]
[end o I loop]
Step 8: for j = 0 to N-1
print A[J]
Step 9: Stop

Page No.
Date / 20

flowchart of bubble Short



* Selection sort :- The selection sort algorithm sorts an array by repeatedly finding the minimum element from the unsorted part and putting it at the begining.

Algorithm for selection sort:

Step 1: START
Step 2: Input N
Step 3: for I = 0 to N-1
Step 4: Read A[I]
Step 5: for I = 0 to N-2
Step 6: S = A[I]
Step 7: pos = I
Step 8: for J = I+1 to N-1
Step 9: If (A[J] < S)
Step 10: S = A[J]
Step 11: pos = J
[end of if]
[end of J for loop]
Step 12: A[pos] = a[I]
Step 13: A[I] = S
[end of I for loop]
Step 14: for I = 0 to N-1
Step 15: print A[I]
Step 16: stop

* Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split one picked and placed at the correct position in the sorted part.

Page No.
Date / 20

* Insertion sort :- Insertion sort is a simple sorting algorithm that builds the final sorted one item at a time by comparisons.

* Algorithm for insertion sort.

Step-1: Start
Step-2: Input N
Step-3: for I = 0 to n-1
Step-4: A[I]
Step-5: J=I
Step-6: while (J>=1) and (A[I]<A[I-1])
Step-7: Temp = A[I]
Step8: A[I] = A[I-1]
Step9: A[I-1] = Temp
Step10: J=J-1
end of while loop
end of step I for loop
step 11: for I = 0 to N-1
Step12: print A[I]
Step13: Stop

* Roots of quadratic equation

(1) Start
(2) Declare the required variables
(3) Read the coefficients a, b, c of Quadratic eq^n
(4) Calculate $d = b^2 - 4ac$
(5) if d<0; display "roots are imaginary" & goto step 6
(6) else calculate $x = (-b + sqrt(d))/2a$
$y = (-b - sqrt(d))/2a$
(6) Display real roots x, y
Stop

* function is a block of code which runs only when it is called.

Page No.
Date / 20

* Time complexity :- Time complexity is the computational complexity that describes the amount of time it takes to run an algorithm.

defined
Q. What is function?
ans ⇒ A function is a group of statement. It divides a large program into small parts. A function can be call multiple time to provide readability.

Types of function

(i) Library function (ii) User Defined function
Predefined function

(i) Library fun :- header file, printf(), main()
pow() - math.h

(ii) User defined function :- That function which is created by user to reduce complexity of a program. (UDF) need to declare it.
Eg :- add(), Mul(), Sub()

```
Void func1();  → function prototype/declaration
Void main()
{
  func1();  → function call
}
Void func1().
{
  ------  → function definition
}
```

Page No.
Date / 20

* Different between User defined function and Library function

	Library fun^n	UDf
(i)	It is already defined in the compiler	It is not predefined in the compiler
(ii)	It is not need to declare	It needs to declare
(iii)	Eg:- printf(), main()	add(), Mul(), sub()
(iv)	These functions are not created by user	These functions are created by user
(v)	Library functions are stored in header file	UDf are not stored in header file
(vi)	The name of library function cannot be changed.	The name user defined function can be changed.

* what is function prototype with example
A function prototype is also known as fun declaration.

Ans ⇒ A function prototype is the declaration of function which contain functions name, parameter and return type.

Ex ⇒ void max(int a, int b);

A function prototype need to be write at the beginning of the program.

Page No.
Date / 20

definition

* what is function ~~call~~ with example

ans ⇒ function definition contains the block of code to perform a specific task.

```
Void max(int a, int b)
{
    if(a>b)
    printf("%d a is greate than b", a, b);
else
    printf("%d a is less than b", a, b);
}
```

function definition

* what is function call with example.

ans ⇒ function call it means that one function calls another function which is user defineed function placed in inside or outside of main function.

Ex ⇒

```
main()
{ a()
  b()
  a()
}
a();
b();
```

```
main()
{ add();
  sub();
}
add()
{ int a,b,c;
  c = a+b;
  printf("%d", c);
}
sub();
{ int a,b,c;
  c = a-b;
  printf("%d", c); }
```

Page No.
Date / 20

WAP to add two number using add(int,int) function.

```
#include<stdio.h>
void main()
void add(int a, int b);
void main()
{
int a=5, int b=6;
add(a,b);
}
void add(int x, int y)
{ printf("Addition is =%d", x+y);
}
output = 11
```

WAP to multiply two number using mul(int, int)

```
#include<stdio.h>
void Mul(int, int);
void main()
{ int a=5;
int b=6;
mul(a,b);
}
void Mul(int x, int y)
{ printf("multiplication is = %d", x*y);
}
output = 30
```

Page No.
Date / 20

D * function code Examples -

0.2 * types of user defined functions with example/ Type

(1) without arguments and without return type
(2) without arguments and with return type
(3) with arguments and without return type
(4) with arguments and with return type

0.3

(1) * WAP to add two number using without argement and without return type.

```
#include <stdio.h>
void add();
int main()
{
 add();
 }
 void add()
{  int a,b,c;
printf("Enter the vallue of a,b");
Scanf("%d %d",&a,&b);
  c=a+b;
printf("the addition of a and b =%d",c);
    }
```

Page No.
Date / 20

0.4

(2) * WAP to add two number without argument and with return type.

```
#include<stdio.h>
int add();
int main()
{ int s;
  s=add();
printf("Sum is = %d", s);
  }

 int add()
{ int a,b,c;
  printf("Enter the value of a,b?");
  scanf("%d %d",&a,&b);
    c=a+b;
  return(c);
    }
```

0.5 (3) * WAP to add two number with argument and with return type:

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

```
#include<stdio.h>
int add (int, int);
int main()
{ int a, b, s;
printf("The value of a, b");
scanf("%d %d", &a, &b);
 s= add(a, b);
printf("the sum = %d", s);
}
int add (int x, int y)
{ int c;
  c = x + y;
  return (c);
  }
```

0.6

Q* WAP to add two number with argument and without return type.

```
#include<stdio.h>
void add (int, int);
int main()
{ int a, b, s;
 printf("Enter the value of a, b");
 scanf("%d %d", &a, &b);
   s = add(a, b);
printf("Sum is = %d", s);
}
void add (int x, int y)
{ int c;
  c = x + y;
}
```

Page No.
Date / 20

* Write a function to add two number and to multiply same number using 4 type of diff-diff functions :-

```
#include<stdio.h>
int add(int, int);
int mul(int, int);

int c;

int add(int a, int b)
{
    c = a+b;
    return(c);
}
int mul(int a, int b)
{
    c = a*b;
    return(c);
}
int main()
{ int a,b,s;p;
printf("Enter the value of a,b");
scanf("%d %d", &a, &b);
    s = add(a,b);
    p = mul(a,b);
printf("sum is %d", s);
printf("multiplication is = %d", p);
    return 0;
}
```

Page No.
Date / 20

* what is actual parameter?

ans ⇒ values that are passed to the called function from the main function are known as Actual Parameter.

* formal parameter : The variables declared in the function prototype or definition are known as formal parameter.

Ex ⇒

```
void main()
int a=5, b=6;
add(a, b); //a and b are the actual parameter in this call.
```

```
void add(int x, int y)
// x, y are formal parameter.
```

* Return statement : If function is returning a value to calling function, it needs to use the keyword return. The called function can only return value per call.

Syntax → return (c);

Page No.
Date / 20

* WAP to find maximum from tow number withargument and with return type.

ans ⇒

```
# include <stdio.h>
int max (int , int);
void main( )
{  int a, b, s;
printf("Enter two number");
scanf("%d %d", &a, &b);

 s = max(a, b);
printf("maximum value is: %d", s);
}

int max (int x, int y)
{
   if (x>y)
   return x;
else
   return y;
   }
```

Page No.
Date / 20

* What is Recursion?

ans⇒ Any function which calls itself is called recursive function and such function calls are called recursive calls.

- R cannot be applied to all problems, but it is more useful.

Ex⇒ factorial

* Working of Recursive function

```
void func();
void main()
{
.....
func();
....
}
void func()
{
.....
func(); → Recursion call
.....
}
```

* Properties of Recursion

- A recursive function can go infinite like a loop.
- To avoid infinite running of recursive function. There are two properties that a recursive function must have.

(i) Base case or Base criteria
it allows the recursion algorithm to stop.

(ii) A base case is typically a problem that is small enough to solve directly.

(2) progressive approach

- A recursive algorithm must change its state in such a way that it moves forward to the base case.

* Properties of recurive function

(i) Base criteria :- There must be at least one condition, such that when this condition is met the function stops calling itself.

(ii) progressive approach :- The recursive calls should progress in such a way that each time a recursive call is made it comes closer to the base criteria.

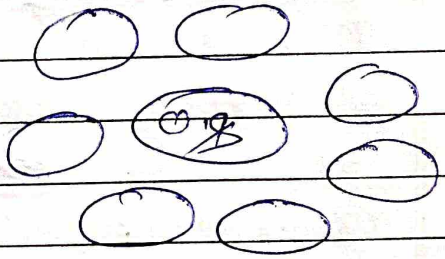
* factorial for function.

Page No.
Date / 20

* Operating System :- Operating system is a system software which interact with hardware, it provids interface between user and machine. Acts as a manager of the Computer system.
Ex⇒ windows, windows 7, windows 8.

* Various functions of operating system.

1. Memory management
2. processor Management
3. Device Management
4. file Management
5. security management
6. Job Scheduling
7. Time sharing etc.



1. Memory management :- Memory management performs the task of allocation and de allocation of memory space to progams. Ex⇒ C Drive, D Drive

2. processor Management :- The operating system assigns different tasks to processors that performed by the Computer system.

3. Device management :- Operating system performs the task of allocating and de-allocating of the devices.

4. file management :- Operating system manages all the files related to activities such as organization, store, naming, sharing and protection of files.

Page No.
Date / 20

(5) Security management :- Security management helps in secure and protect the computer system internally as well as externally.

(6) Job management :- Job management performs of allocating of different-different Jobs tasks by the operating system.

(7) Time sharing :- Time sharing co-ordinate and assign compilers, assemblers, programs and other software to various users working on computer system.

Page No.
Date / 20

* What is linear search?

ans> Linear search also known as sequential search, It is a process that checks every element in the list sequentially until the desired element is found.

Program

```
#include<stdio.h>
int main{
int arr[50], size, element, i;
printf("Enter your size of array");
scanf("%d",&size);
printf("Enter your an array element:");
[redacted]
for(i=0; i<size; i++)
 { scanf("%d",&arr[i]);
 }
printf("Enter an array element to be search:");
 scanf("%d",&element);
for(i=0; i<size; i++)
{
  if(element==arr[i])
{ printf("Element is found at index:%d",i);
  break;
}}
if(i==size)
{printf("Element is not present in the array");
 return 0;
 }
```

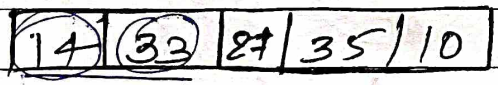
Page No.
Date / 20

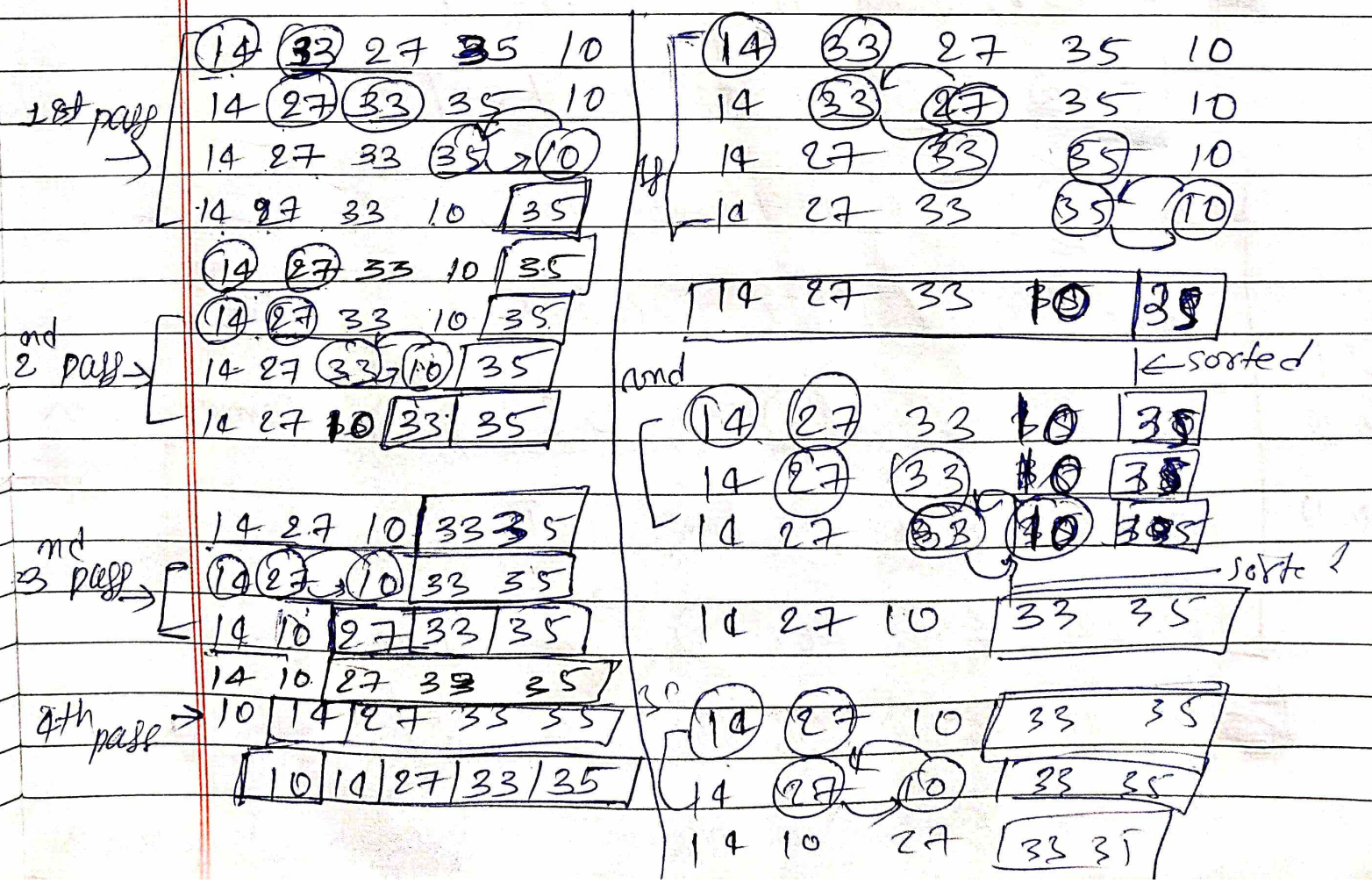
* Binary search :- Binary search also known as half interval search, binary search is a sorting algorithm, that finds an element and position as well as index in a sorted array. Binary search technique works only on a sorted array. Binary search compares the target value to the middle element of the array.

```
#include<stdio.h>
 void binarysearch(int a[], int s, int ele)
{ int i=0, f=s-1, m;
while(i<=f)
{  m=(i+f)/2;
   if(ele==a[m])
   { printf("Element is found at index: %d\n",m);
     break;
     }
else if(ele>a[m])
{  i=m+1; }
else if(ele<a[m])
{  f=m-1; }
  }
else { printf("element is not present"); }
  }
int main()
{ int ele, a[1000], s, i;
  printf("Enter your size of array: ");
   scanf("%d",&s);
printf("Enter your array element:\n");
   for(i=0; i<s; i++)
   { scanf("%d",&a[i]); }
```

```
printf("Enter an array element to be
        search:");
scanf("%d",&ele);
binarysearch(a,s,ele);
return 0;
}
```

* Bubble sort :- Bubble sort is a simple sorting algorithm that compares two adjacent element and swaps them repeatedly if they are in wrong order. bubble sort arrange the array in asending order. It requires (n-1) pass to sort. Time complexity is $O(n^2)$





Page No.
Date / 20

* 55, 66, 33, 22, 77, 11, 88, 12

1st pass →

55 66 33 22 77 11 88 12

55 66 33 22 77 11 88 12

55 33 66 22 77 11 88 12

55 33 22 66 77 11 88 12

55 33 22 66 77 11 88 12 → 7 Comparison

55 33 22 66 11 77 88 12

55 33 22 66 11 77 88 12

55 33 22 66 11 77 12 [88] sorted

2nd pass (n-1)

55 33 22 66 11 77 12 [88]

33 55 22 66 11 77 12 [88]

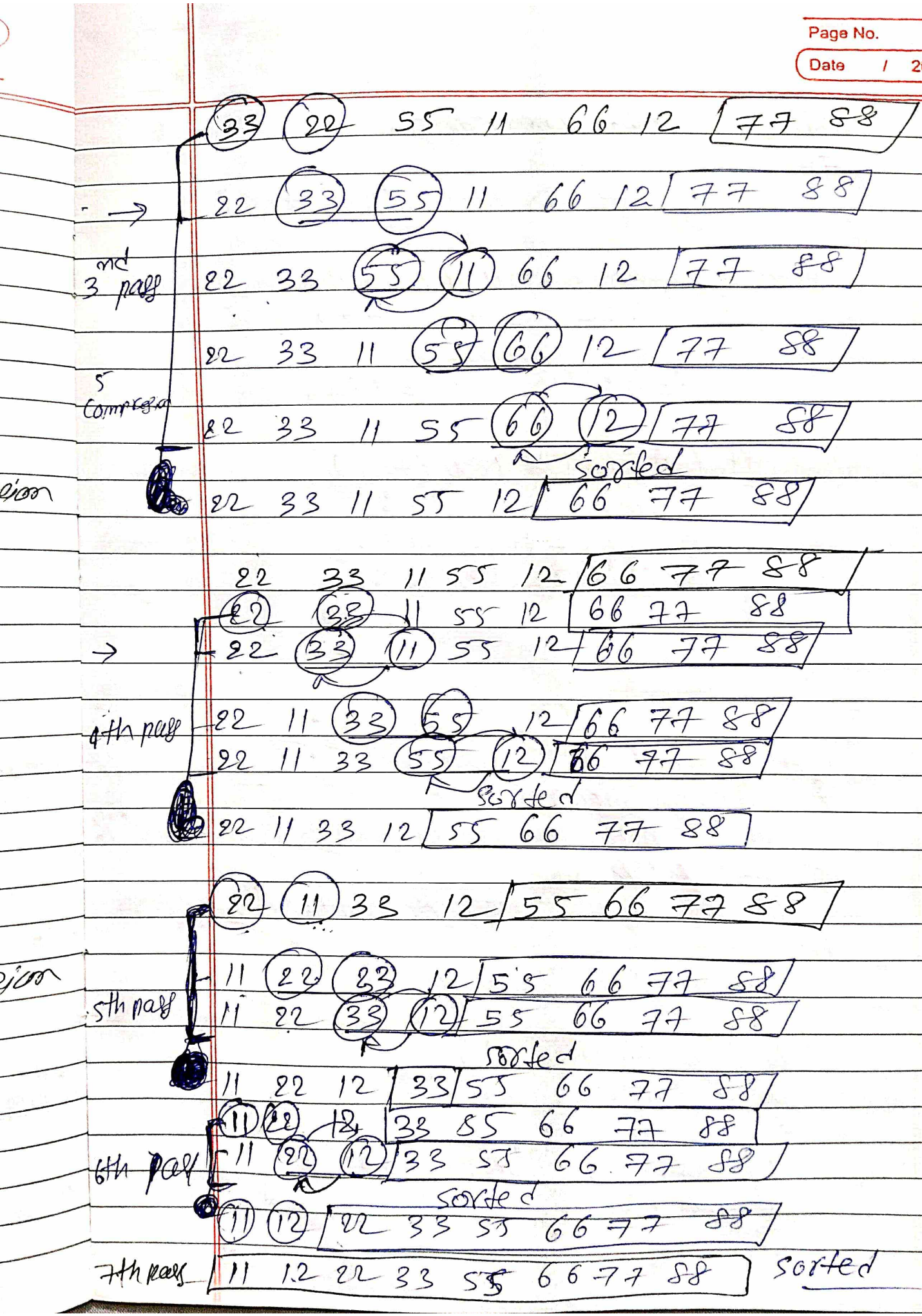
33 22 55 66 11 77 12 [88]

33 22 55 66 11 77 12 [88] → 6 Comparison

33 22 55 11 66 77 12 [88]

33 22 55 11 66 77 12 [88]

33 22 55 11 66 12 [77] [88] sorted


33 22 55 11 66 12 77 88
22 33 55 11 66 12 77 88
3rd pass
22 33 55 11 66 12 77 88
22 33 11 55 66 12 77 88
5 Comparison
22 33 11 55 66 12 77 88
sorted
22 33 11 55 12 66 77 88
22 33 11 55 12 66 77 88
22 33 11 55 12 66 77 88
22 33 11 55 12 66 77 88
4th pass
22 11 33 55 12 66 77 88
22 11 33 55 12 66 77 88
sorted
22 11 33 12 55 66 77 88
22 11 33 12 55 66 77 88
5th pass
11 22 33 12 55 66 77 88
11 22 33 12 55 66 77 88
sorted
11 22 12 33 55 66 77 88
11 22 12 33 55 66 77 88
6th pass
11 22 12 33 55 66 77 88
sorted
11 12 22 33 55 66 77 88
7th pass
11 12 22 33 55 66 77 88
sorted

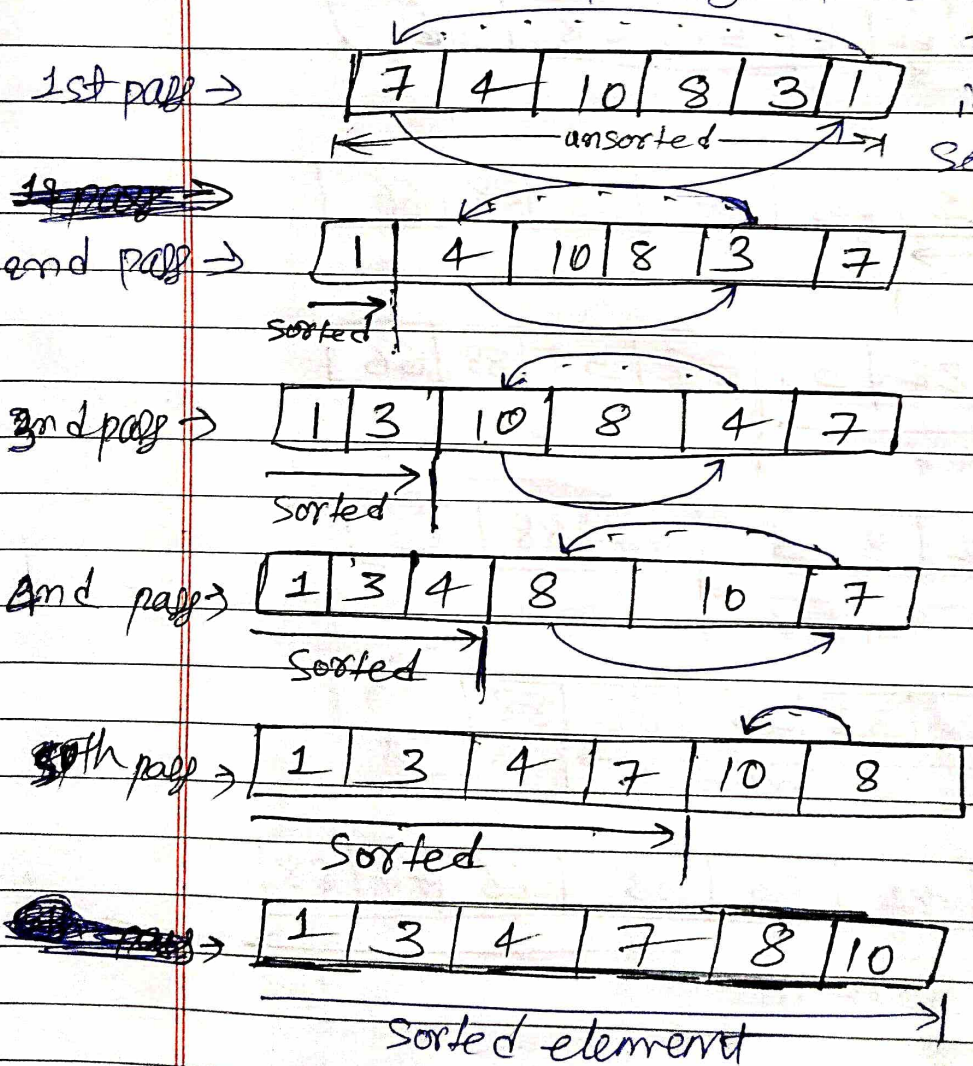
* Bubble sort program.

```
#include<stdio.h>
void bubblesort(int arr[], int s)
{
  int i, temp, p;
  for(p=0; p<=s-1; p++)
  {
    for(i=0; i<s-1-p; i++)
    {
      if(arr[i] > arr[i+1])
      { temp = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = temp;
      }
    }
  }
}
int main()
{ int s, arr[100], i;
printf("Enter the size of your array element:");
scanf("%d", &s);
printf("Enter your array element:\n");
for(i=0; i<s; i++)
  { scanf("%d", &arr[i]);
  }
  bubblesort(arr, s);
  for(i=0; i<s; i++)
  {
  printf("%d ", arr[i]);
  return 0;
  }
```

Page No.
Date / 20

* Selection Sort :- Selection Sort is an effective and efficient sorting algorithm that finds repeatedly the minimum element from the unsorted array and put it at the begining of an array. Selection sort arrange the array elements in asending and as well as desending order. Time complexity of SS is $O(n^2)$

the no of pass in selection sort is $(n-1)$.

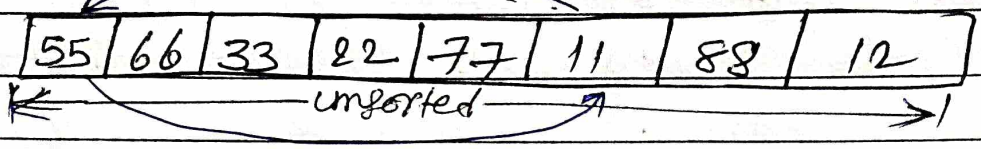


Page No.
Date / 20

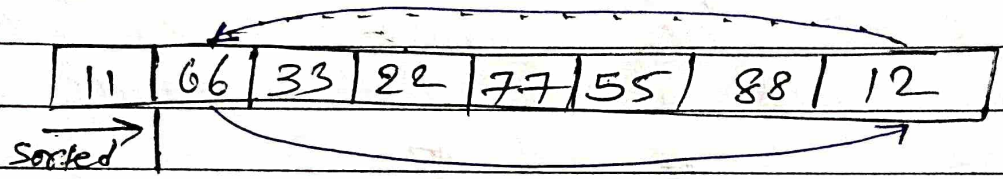
D → 0.8

* 55, 66, 33, 22, 77, 11, 88, 12

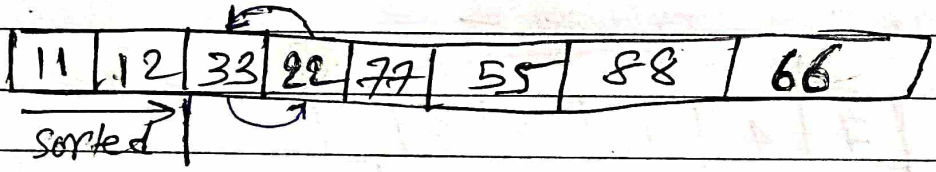
1st pass →



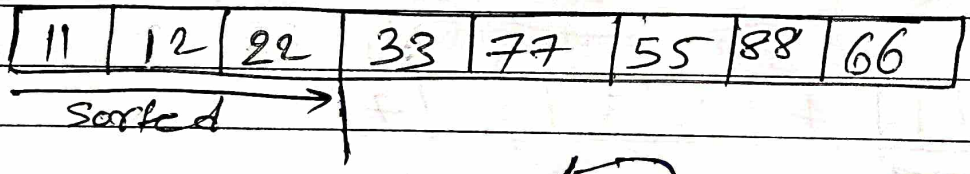
2nd pass →



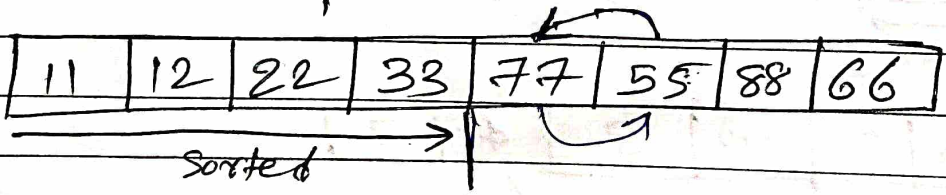
3rd pass →



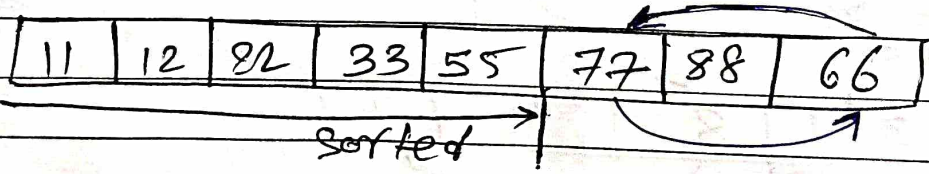
4th pass →



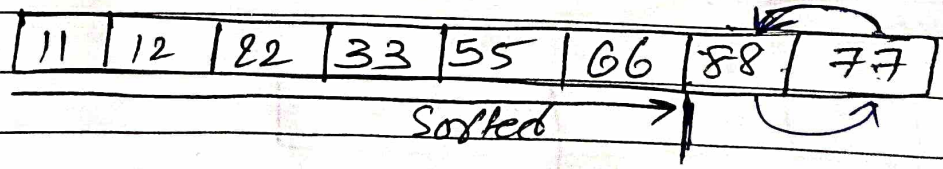
5th pass →

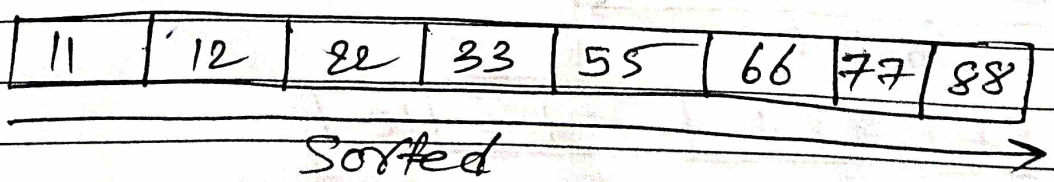


6th pass →



7th pass →





Page No.
Date / 20

* Selection sort program:

```
#include<stdio.h>
void selectionsort(int arr[], int s)
{ int min, i, j, temp;
  for(i=0; i<s-1; i++)
  { min=i;
    for(j=i+1; j<s; j++)
    { if(arr[j]<arr[min])
      { min=j;
      }
    }
    if(min!=i)
    {
      temp=arr[i];
      arr[i]=arr[min];
      arr[min]=temp;
    }
  }
}
int main()
{ int s, arr[200], i;
  printf("Enter your array size:");
  scanf("%d",&s);
  printf("Enter your array elements:\n");
  for(i=0; i<s; i++)
  {
    scanf("%d",&arr[i]);
  }
  Selectionsort(arr, s);
  for(i=0; i<s; i++)
  { printf("%d ", arr[i]);
  }
  return 0;
}
```

Page No.
Date / 20

* Different between bubble sort and selection sort.

	Bubble sort	selection sort
(i)	Bubble sort is a simple sort algorithm that Compares two adjacent element and swaps them.	Selection sort is an effective and efficient sorting algorithm that finds the minimum element from the unsorted array and put it at the begining of an array.
(ii)	it can arrange array elements in only asending order.	it can arrange array elements in asending as well as desending order.
(iii)	It Compares the adjacent elements and swap	It finds the minimum element in the unsorted array.
(iv)	Less efficient	More efficient
(v)	slower	faster

* Selection sort is better than bubble sort, How?

ans → Selection sort performs a small number of swaps Compared to bubble sort, Selection sort performs faster and more efficiently.

Date / 20

* Insertion sort :- Insertion sort is a simple sorting algorithm that builds the final ~~the final~~ sorted array by transferring one element at a time. time complexity is $O(n^2)$

12, 45, 23, 51, 19, 8

1st pass → | 12 | 45 | 23 | 51 | 19 | 8 |

sorted : unsorted →

2nd pass → | 12 | 45 | 23 | 51 | 19 | 8 |

3rd pass | 12 | 23 | 45 | 51 | 19 | 8 |

4th pass | 12 | 23 | 45 | 51 | 19 | 8 |

5th pass | 12 | 19 | 23 | 45 | 51 | 8 |

6th pass | 8 | 12 | 19 | 23 | 45 | 51 |

sorted →

* 55, 66, 33, 22, 77, 11, 88, 12

| 55 | 66 | 33 | 22 | 77 | 11 | 88 | 12 |

sorted : → unsorted

| 55 | 66 | 33 | 22 | 77 | 11 | 88 | 12 |

sorted →

| 33 | 55 | 66 | 22 | 77 | 11 | 88 | 12 |

| 22 | 33 | 55 | 66 | 77 | 11 | 88 | 12 |

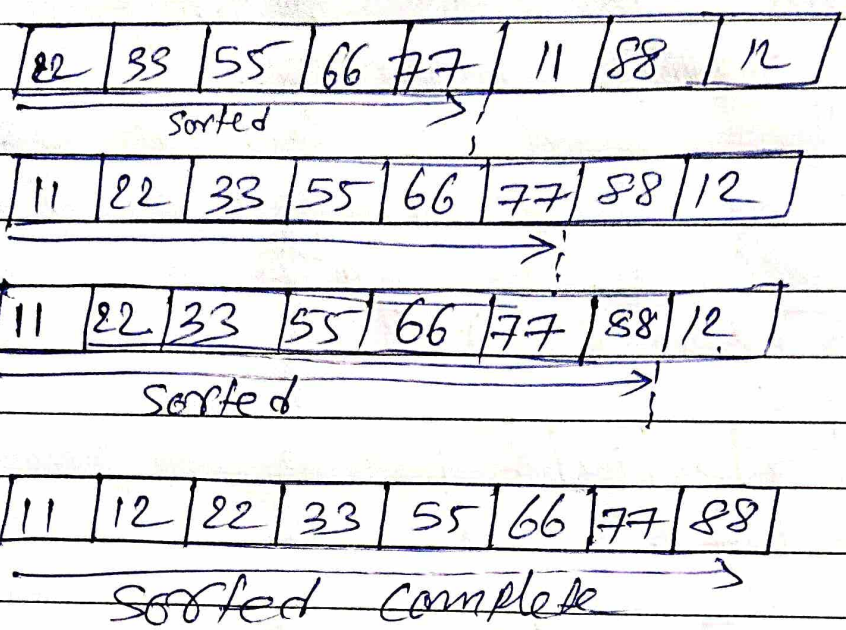
sorted →

| 22 | 33 | 55 | 66 | 77 | 11 | 88 | 12 |

sorted →

Page No.
Date / 20

Repeate



*

Insertion sort	Selection sort
(i) Insertion sort is a simple sorting algorithm that builds the final sorted array by transferring one element at a time.	Selection sort is effective sorting algorithm that finds the minimum element from the unsorted array and put it at beginings of an array.
(ii) It transforme an element at a time.	It finds minimum element from array and swap them.
(iii) It is more efficient than selection sort.	It is less efficient than insertion sort.
(iv) It is complex than selection sort.	It is simpler than insertion sort.

Page No.
Date / 20

* Insertion sort program

```
#include<stdio.h>
void insertion_sort(int arr[], int s)
{ int i, j, temp;
 for(i=1; i<s; i++)
{ temp= arr[i];
   j= i-1;
 while(j>0 && arr[j]>temp)
 { arr[j+1]= arr[j];
    j--;
 }
   arr[j+1]= temp;
}
}
 int main()
{ int s, arr[200], i;
 printf("Enter your array size:");
 scanf("%d",&s);
printf("Enter your array elements:\n");
 for(i=0; i<s; i++)
 { scanf("%d",&arr[i]);
 }
  insertion_sort(arr, s);
  printf("your sorted array is:\n");
   for(i=0; i<s; i++)
   { printf("%d ", &arr[i]);
   }
  return 0;
 }
```

Page No.
Date / 20

* Discuss the types of errors with example.

(1) Syntax error
(2) Semantic error
(3) Logical error
(4) Linker error
(5) Runtime error

(1) Syntax error :- Syntax error occurs when you do mistakes in writting of source code and also when you break the rule of writting in C are known as Syntax error.

Ex ⇒
```
#include<stdio.h>
int main()
{ a=10;
printf("%d",a);
return 0;
}
```
output → Syntax error because a is undeclared.

↳ Syntax error does not compile successfully compiler gives error.

(ii) Semantic error :- Semantic error occur when text of source code is grammatically correct but doesn't make any sence. It is not compile successfully, compiler gives error.

Eg ⇒
```
{ int a,b,c;
a=10;
b=25;
a+b=c;
}
```

Page No.
Date / 20

(iii) logical error :- In this error source code compiles successful and compiler does not gives any error but we do not get desire output of our sourcecode.

Ex →
```
#include<stdio.h>
Void main()
{  int a = 8;
   int b = 3;
   int c;
c = a / b;
printf("%d", c);
}
```

output → 2 (Not desire value)
desire value is 2.666)

Solution → a, b, c should be declared with float data type and format Specifire is %f . then output is 2.666.

(iv) Linker error :- Linker error occurs when any text error in predefined function . If does not compile successfully, Compiler gives error in Linker error.

Ex →
```
#include<stdio.h>
Void main()
{ print("Rajthan");
}
```

Output → error → undeclared symbal.

Page No.
Date / 20

Ans: Runtime error : Runtime error occurs during running of source code or program, it compile successfully but we donot get any output.

```
#include<stdio.h>
void main()
{int a = 10;
int b = 0;
int c;
c = a / b;
printf("c=%d",c);
}
```

Ans: write a program to add 2 number using pointer.

```
#include<stdio.h>
void add(int *a, int *b)
{ int c;
  c=*a+*b;
printf("Addition of two number is %d",c);
}
void main()
{ int a,b;
  a=58, b=55;
add(&a,&b);
}
```

Page No.
Date / 20

6 ** add two matrix (3x3) order

```
#include<stdio.h>
void main()
{ int a[3][3], b[3][3], c[3][3], i, j;
  for(i=1; i<=3; i++)
  { for(j=1; j<=3; j++)
    {printf("\nfirst matrix: a%d%d=", i, j);
     scanf("%d", &a[i][j]);
    }
  }
for(i=1; i<=3; i++)
  {
   for(j=1; j<=3; j++)
    {printf("\nsecond matrix: b%d%d=", i, j);
     scanf("%d", &b[i][j]);
    }
  }
for(i=1; i<=3; i++)
  { for(j=1; j<=3; j++)
    { c[i][j]=a[i][1]+b[i][1];
    }
  }
for(i=1; i<=3; i++)
  { for(j=1; j<=3; j++)
    { printf("%d\t", c[i][j]);
    }
    printf("\n");
  }
}
```

Page No.
Date / 20

* Do while ka Code :

```
#include<stdio.h>
  int main()
{ int n=1;
   do
   {
   printf("%d\n",n);
     n++;
   }
  while(n<=5);
   return 0;
    }
```

* while ka Code

```
#include<stdio.h>
 void main()
{  int n=1;
  while(n<=5)
   {
    printf("y.d\n",n);
    n++;
    }
```

* Do while :- Do while is a variable of the while loop. This loop executes atleast one time before checking the condition and then if condition is true then it will repeat the loop as long as the condition is true.

Page No.
Date / 20

* Differentiate between while and do while loop.

	while	Do-while
(i)	while loop checks the condition first and then execute the statement.	Do while loop executes the statements first and then checks the condition.
(ii)	while loop executes the statement only if the condition is true	Do while executes the statement at least one time even if the condition is false.

(iii)	syntax:
	while (condition)
	{
	// statement
	}
		syntax:
		do
		{ // Statement
		} while (condition);

(iv)	No semi colon is used in syntax of while (condition).	Semi colon is used in syntax of while (condition);

* write the syntax for 'for' loop.

ans → for (initial variable; condition; increment)
{
// statements;
}

Page No.
Date / 20

* Explain the use of scanf function.

ans ⇒ scanf function is predefined function in header file compiler well known about scanf function, scanf function is use to take input from user. through scanf function we can give integer, float, character, string, array input to perform a specific task.

* write a program to make a pyramid pattern using numbers and the output of your program must exactly match the below given.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
#include<stdio.h>
int main()
{ int i,j;
 for(i=1;i<=5;i++)
  {
   for(j=1;j<=i;j++)
    {
     printf("%d ",j);
    }
   printf("\n");
  }
 return 0;
}
```

the general form of a pointer variable declaration is ⇒ data type * name

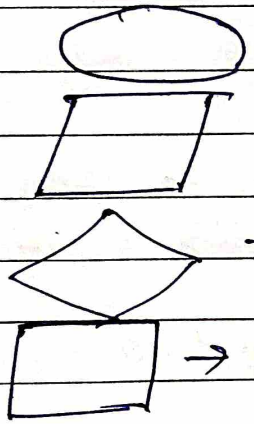
Ex ⇒ int *a;

Page No.
Date / 20

* Define flowchat

→ flowchat is the diagrammatic representation of an algorithm in flowchat some specific symbols are used to solving a task through flowchat algorith of programe becoms easy to understend.

Ex=



→ Start/End

→ input and output

→ Condition

→ process

* what is algorithm?

any → An algorithm is a set of statements and commands that must be followed for a computer to perform calculations or other operations.

or → An algorithm is a fine set of instructions carried out in a specific order to perform a particulary task.

* pointer: A pointer is a variable which stores the address of another variable and it's value.

```
int a = 10;
int * pr;
pr = &a;
```

printf("address of pr = %u", pr);

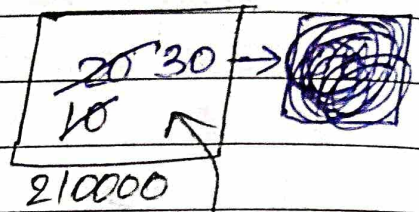
Page No.
Date / 20

* Demonstrate the working of a pointer with suitable example :

Example : working

ans)

```
int a = 10;
int *ptr;
*ptr = &a;
*ptr = 20;
int **ptr = &ptr;
**ptr = 30;
```



* what are functions? List types of user defined functions with suitable example.

ans) Call > 0.1, 0.2, 0.3, 0.4, 0.5, 0.6

* Distinguish between searching and sorting any sorting algorithm by taking a suitable example.

	Searching	Sorting
(i)	It is the process of checking element is present in the array or not.	It is the process of arranging the element in ascending or descending order in the array.
(ii)	Example of searching : linear search, binary search	Example of Sorting : bubble sort, Selection sort insertion sort.
(iii)	It searches the element given by user and comparing with array element until the desire value is found.	It sorts the given elements by user in ascending and descending order.

Page No.
Date / 20

* by taking a suitable example sorting call 0.8

* Illustrate the need of recursion in C.

ans ⇒ Recursion technique provides a way in C language to break complicated problems into simple problems which are easier to slove.
Recursion can be used in case of similar subtask like sorting, searching, and traversal problems.
Example ÷ sum of Numbers using recursion

```
#include<stdio.h>
int add(int n)
{ int d;
 if(n<=1)
  { return n; }
 else {
        d=n+add(n-1);
        return d;
      }
}
  int main()
{ int s,n;
 printf("Enter a number:");
  scanf("%d",&n);
  s=add(n);
  printf("Sum of n numbers is:%d",s);
  return 0;
  }
```

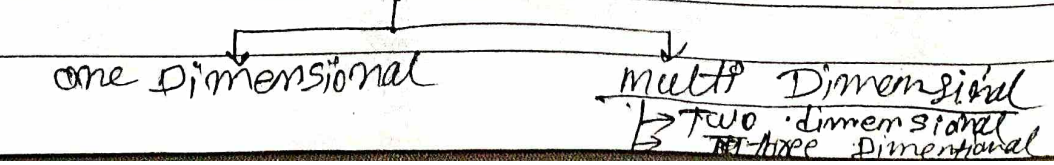
Page No.
Date / 20

* factorial using recursion

```
#include<stdio.h>
int fact(int n)
{ int d;
  if(n==0)
   { return 1;
   }
  else {
       d = n * fact(n-1);
       return d;
      }
}
int main()
{  int n, s;
printf("Enter a number:");
scanf("%d",&n);
 s= fact(n);
printf("the factorial of given number is %d",s);
  return 0;
   }
```

* Define Array :- Array is the collection of similar data types stored at contiguous memory location. It is a variable which can store multiple values.

```
int main()
{  int array[100];
}
```



Page No.
Date / 20

* WAP in C to print the first 50 natural numbers using recursion.

ans =>
```
#include<stdio.h>
int add(int n)
{
  if(n<=49)
  {
  printf("%d",n);
  add(n+1);
  }
}
int main()
{ int n=1;s;
  s=add(n);
  printf("%d\n",s);
  return 0;
  }
```

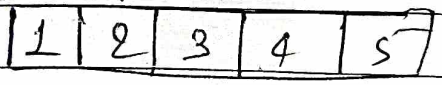
* WAP to display address of a variable using pointer.

ans =>
```
#include<stdio.h>
void main()
{ int a;
  int *ptr;
  ptr=&a;
printf("the address of a=%u",ptr);
  }
```

Page No.
Date / 20

* Difference between 1D-array and 2-D array

ans ⇒

	1-D array	2-D array
(i)	1D array contains single row and multiple columns.	2D array contains multiple row and multiple columns.
(ii)	1D array is the collection of similar data types stored at contiguous block of memory location	2D array is the collection of similar data types stored at like table and Matrix of memory location.
(iii)	It is also called single dimensional array	It is also called Multi-dimensional array
(iv)	Stores data as a list	Stores data as in row-column format.
(v)	Example → 1D array 	Example : 2D array $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
(vi)	Syntax :- data type - name[size] = {elements};	Syntax :- data type - name[row][column];

* which operator is used to compare two variable.

ans ⇒ Relational operator.

* print the elements of a 2D array

* How to multi dimension array are declared in C:

ans ⇒

```
#include<stdio.h>
void main()
{ int i,j;
int Rav[3][3]={1,2,3,4,5,6,7,8,9};
for(i=0; i<3; i++)
{ for(j=0; j<3; j++)
  {
    printf("%d\t", Rav[i][j]);
  }
  printf("\n");
}
}
```

* Call by value :- Call by value ~~method~~ is the method of passing value to formal parameter from actual parameter is called call by value. In this method ~~when we~~ the copy of variable is passes to the user defined function so if there is change with variable in the user defined function than there is no any effect and no any change in actual parameter of the main function

working

```
void main()
{ int x=10, y=20;
swap(x,y);     [10] [20]
printf("%d %d", x,y);
```

x = 10
y = 20

```
void swap(int x, int y)
{ int t = x;
  x = y;
  y = t;
printf("%d%d", x,y);
```

x: [~~20~~ 10] → 20, y: [~~10~~ 20] → 10

x = 20
y = 10

Page No.
Date / 20

Call by value

```
#include<stdio.h>
void swap(int a, int b)
{   int temp;
    temp = a;
    a = b;
    b = temp;
printf("the value of a is %d\n", a);
printf("the value of b is %d\n", b);
}

  void main()
{   int a, b;
printf("Enter the value of a, b:");
  scanf("%d %d", &a, &b);
printf("the value of a before swap %d\n", a);
printf("the value of b before swap %d\n", b);
  swap(a, b);
printf("the value of a after swap %d\n", a);
printf("the value of b after swap %d\n", b);
}
```

Call by reference :- Call by reference is the method of passing address of variable to formal parameter from actual parameter is called call by reference. In this method the address of variable is passes to the user defined function so if there is any change with variable in the user defined function than there will be effect and change in actual parameter of the main function.

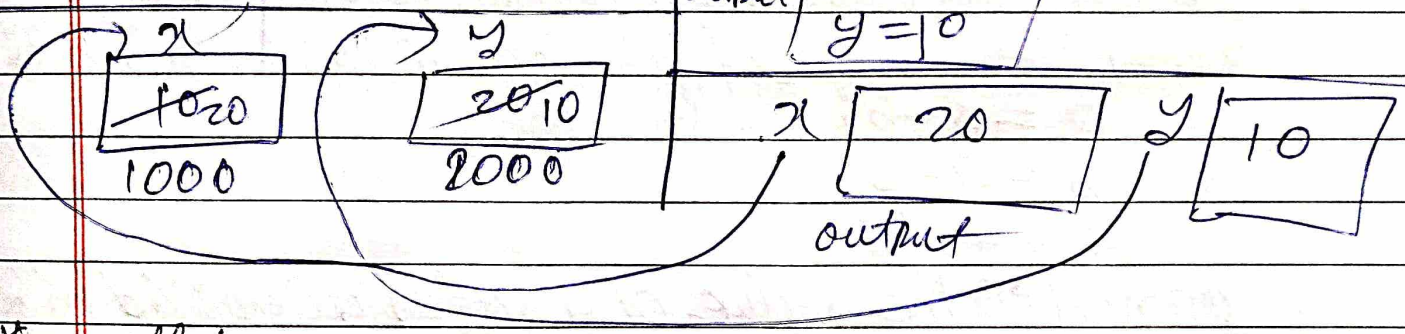
Working

```
void main()
{ int x=10;
    y=20;
swap(&x,&y);
printf("%d %d",x,y)
```

out = x=20
y=10

```
void swap(int*x,int*y)
{ int temp;
temp=*x;
*x=*y;
*y=temp;
printf("%d %d",*x,*y);
```

output x=20 y=10



* call by Reference :

```
#include<stdio.h>
void swap(int *a, int *b)
{ int temp = *a;
  *a = *b;
  *b = temp;
}
Void main()
{ int a,b;
printf("enter the value of a,b");
scanf("%d %d",&a,&b);
printf("the value of a before swap %d\n",a);
printf("the value of b before swap %d\n",b);
  swap(&a,&b);
printf("the value of a after swap %d\n",a);
Printf("the value of b after swap %d\n",b);
}
```

Page No.
Date / 20

* WAP to swap two number without using three variable.

Sol:
```
#include<stdio.h>
int main()
{ int a,b;
printf("Enter the value of a & b:");
scanf("%d%d",&a,&b);
printf("the value of a before swap:%d\n",a);
printf("the value of b before swap:%d\n",b);
    a=a+b;
    b=a-b;
    a=a-b;
printf("the value of a after swap:%d\n",a);
printf("the value of b after swap:%d\n",b);
    return 0;
    }
```

* WAP to add ten numbers in array.

```
#include<stdio.h>
Void main()
{ int i, sum=0, arr[500], c;
printf("Enter your 10 array element:\n");
  for(i=0;i<=9;i++)
  { scanf("%d",&arr[i]);
  }
for(i=0;i<=9;i++)
  { sum=sum+arr[i];
  }
printf("the sum of term array ele-i&:%d",sum);
  }
```

Page No.
Date / 20

* Difference between printf() and puts().

ans ⇒

	printf()	puts()
(i)	It is use for display the output	It is also use for display the output
(ii)	printf() prints string, digit, character	puts prints only string.
(iii)	printf() does not creates new line automatically	puts creates automatically new line.
(iv)	for new line we have use '\n'.	for new line we do not use '\n'.

* Difference between flowchat & algorithm

	flow chat	algorithm
(i)	flowchat is the graphical representation of program.	algorithm is step by step description in terms of alphabet.
(ii)	for flowchat we have to use special symbols and notations	for algorithm we have to use simple english language
(iii)	It is more time consuming	It is less time consuming
(iv)	Difficult to modify	Easy to modify.
(v)	difficult to debug error	Easy to debug error.

Page No.
Date / 20

* Difference between Compiler and Interpreter.

	Compiler	Interpreter
(i)	Compiler produce not only binary code but also produce intermediate code	Interpreter produce only Binary code
(ii)	Compiler translate all line together.	Interpreter does not translate all lines together rather it translate line by line.
(iii)	Compiler show all errors at written time	It show only one error at written time.
(iv)	less time to execute code	more time to execute code
(v)	Debuging is slow	Debuging is fast.

Page No.
Date / 20

☆ difference between primary memory and secondary memory and Ram and Rom.

	primary memory (RAM)	secondary memory (ROM)
(i)	It stands fo Random access memory	It stands for Read only memory.
(ii)	It is volatile	It is non volatile.
(iii)	It contain data as long as the computer is on	Computer is on or off no matter it contains data for lifetime.
(iv)	It can be modify.	It can't be modify.
(v)	It works fast than secandary memory or ROM.	It works slow than primary memory and RAM
(vi)	It is more expensive.	It is cheap than RAM and primary memory.
(vii)	Example :- (i) DRAM (ii) SRAM	Example :- SSD, HDD.

☆ Precidence operator decision karta है the all operators के लिए।

* Datatype, name of primary datatype with their memory require~~ment~~

Page No.
Date / 20

* What is Datatype?
Data type is the declaration of variable ~~that~~ which tells the Compiler what kind of variable we want to take integer, float, character.

* Types of Data type

- Primary Datatype or primitive datatype
  - Int
  - float
  - char
  - Void
- Derived Data type or secondary datatype
  - Array
  - string
  - pointer
- User defined datatype
  - structure
  - union
  - Typedef
  - enum

memory size of data type

int — 2 bytes
float — 4 bytes
char — 1 bytes
Double — 8 bytes

* WAP to print pattern like right angle triangle with number increasing by 1.

```
#include<stdio.h>
void main()
{ int i, j, no=0;
for(i=1; i<=4; i++)
{ for(j=1; j<=i; j++)
  { no++;
    printf("%d ", no);
  }
printf("\n");
}
}
```

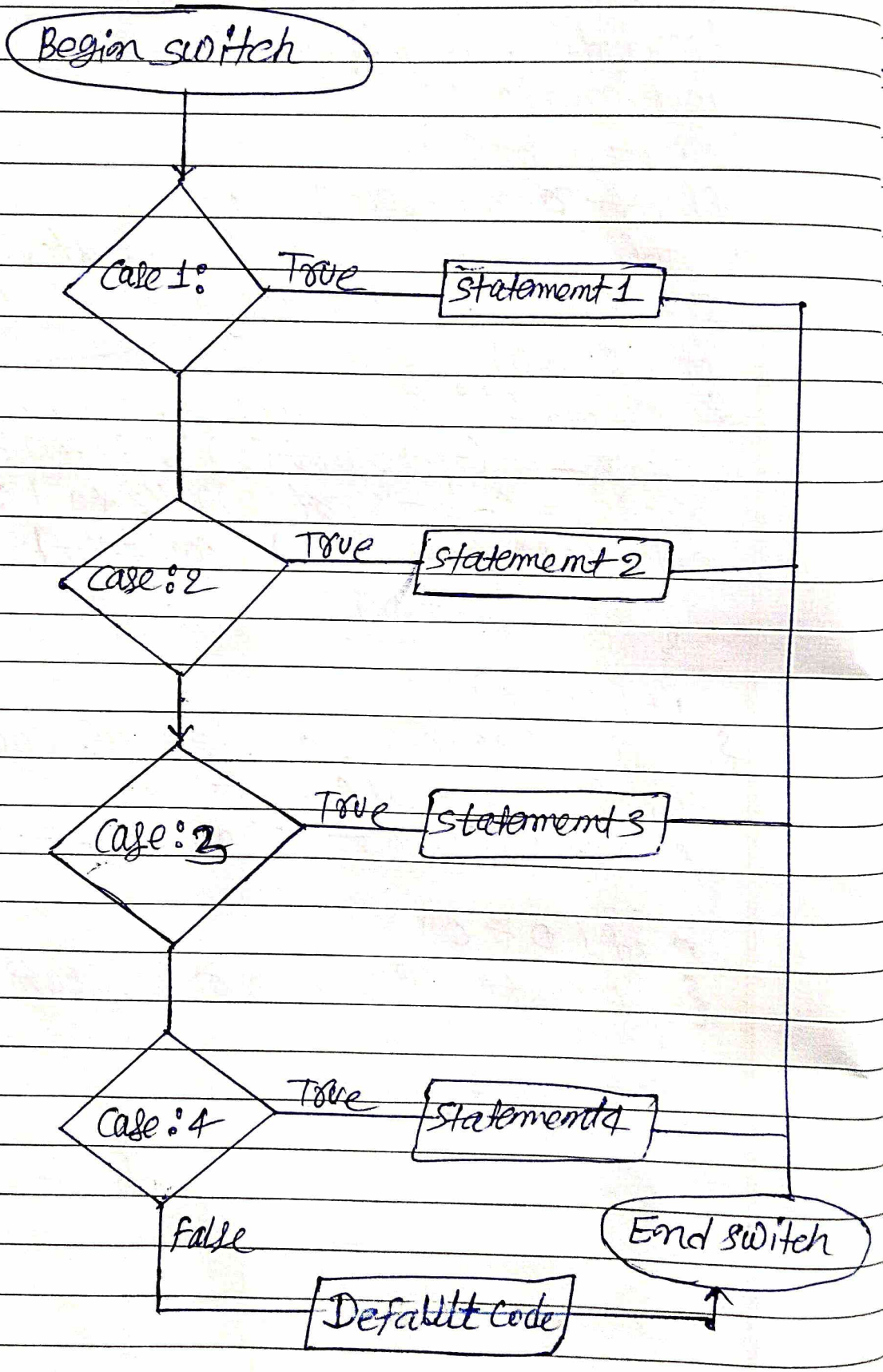
Page No.
Date / 20

* WAP to find roots of a quadratic eqn.

```
#include<stdio.h>
#include<math.h>
int main()
{ int a,b,c;
float root1,root2,D;
printf("Enter the coefficient of a,b,c\n");
D= b*b - 4*a*c;  scanf("%d%d%d",&a,&b,&c);
if(D>0)
{ printf("real and unequal roots\n");
root1=(-b+sqrt(D))/(2*a);
root2=(-b-sqrt(D))/(2*a);
printf("roots are\nroot1=%f\nroot2=%f",root1,
        root2);
}
if(D=0)
{ printf("real and equal roots\n");
root1=-b/(2.0*a);
printf("root is\n root1=%f",root1);
}
if(D<0)
{ printf("Imaginary roots\n");
}
return 0;
}
```

Page No.
Date / 20

* flowchat for switch case



Page No.
Date / 20

Define

* structure :- structure is a user defined datatype. It is a collection of similar and dissimilar elements. It is used to store data of student, employee etc. with using only one variable with the help of structure, it stores different type of daty like :- int type, float type, char type.

* How to declare a structure variable
or - Syntax of structure

```
struct structure tag
{
    member 1;
    member 2;
    member 3;
    member n;
};
```

keyword → struct, tag

Example:

```
struct student
{
    char name[20];      // member 1
    char college name[50]; // member 2
    int roll;           // member 3
    float marks;        // member 4
};
```

body ←

Page No.
Date / 20

* difference between array and structure.

	array	structure
(i)	array is the collection of similar data type stored at contigous memory location.	structure is the collection of dissimilar data type stored at contigious memory location.
(ii)	array has no any key-word.	It has a key word that is 'struct'.
(iii)	array takes less time than structure	It takes more time than array.
(iv)	Array uses square bracket "[ ]" for access the elements.	structure uses "." Dot operator for access the elements.
(v)	array is a pointer	structure is not a pointer.
(v)	syntax :- data type-name[size];	syntax :- struct tag { member1; member2; membern; };

* How structures are different from an array :-

ans ⇒ structures can be used as container which can hold variables of different data type on the other hand Array is used as container which can hold variables of same data type, it does not contain different data type.

Page No.
Date / 20

* Define array of structure: An array of structure is simply an array in which each element is a structure of the same data type.

* Define array of structure: An array of structure is a collection of different datatype variables, grouped together under a single name.

```
#include<stdio.h>
struct employee
{ int emp_id;
  char name[20];
  float salary;
};
int main()
{ int i,n;
  struct employee d[10];
  for(i=0; i<=9; i++)
  { printf("Enter details for employee:%d\n
  employee id:\n name:\n salary:\n", i+1);
  scanf("%d", &d[i].emp_id);
  scanf("%s", d[i].name);
  scanf("%f", &d[i].salary);
  }
  printf("Enter employee id to search:");
  scanf("%d", &n);
  for(i=0; i<=9; i++)
  { if(n==d[i].emp_id)
```

{printf("employee id is: %d \n", d[i].emp-id);
printf("employee name is: %s \n", d[i].name);
printf("employee salary is: %f \n", d[i].salary);
return 0;
}
}

printf("Data not valid");
return 0;
}

* Define string:- string is a sequence of characters and string is a array of character terminated with a null character '\0'. EX⇒ syntax→ char str[];

* WAP to find out length of a string:-

```
#include<stdio.h>
void main()
{int i;
char a[25];
printf("Enter a string:");
scanf("%s", a);
for(i=0; a[i]!='\0'; i++);
printf("Length of a string is: %d", i);
}
```

Page No.
Date / 20

* Create a structure "Employee" having EmpID, Name, Salary and Age as its members.

```
#include<stdio.h>
Struct employee
{ int emp-id;
  char name[20];
  float salary;
  int age;
};
int main()
{ Struct employee x;
printf("Enter employee id:\n");
Scanf("%d",&x.emp-id);
printf("Enter employee name:\n",);
scanf("%s",x.name);
printf("Enter employee salary:\n");
Scanf("%f",&x.salary);
printf("Enter employee age:\n");
Scanf("%d",&x.age);
printf("employee details shown below:\n");
printf("employee id is:%d\n",x.emp-id);
printf("employee name is:%s\n",x.name);
printf("employee salary is:%f\n",x.salary);
printf("employee age is:%d",x.age);
return 0;
}
```

Page No.
Date / 20

* Table of 3

```
#include<stdio.h>
int main()
{ int a=3,c,x;
  for(x=1;x<=10;x++)
  { c=a*x;
    printf("%d\n",c);
  }
  return 0;
}
```

* Create a ~~Student~~ structure "student" with roll no, Name, Branch, and Marks its members;

```
#include<stdio.h>
  struct student
{  int roll;
   char name[50];
   char branch[50];
   float marks;
   char college[50];
};
  int main()
{ struct student s;
printf("Enter student roll:\n");
scanf("%d",&s.roll);
printf("Enter student Name:\n");
scanf("%s",s.name);
printf("Enter student branch:\n");
scanf("%s",s.branch);
printf("Enter student marks:\n");
scanf("%f",&s.marks);
printf("Enter student college name:\n");
  scanf("%s",s.college);
```

Page No.
Date / 20

printf("The information of student shown below:\n");
printf("Roll no: %d\n", s.roll);
printf("Name: %s\n", s.name);
printf("Branch: %s\n", s.branch);
printf("Marks: %.2f\n", s.marks);
printf("College Name: %s\n", s.college);
return 0;
}

* Create a structure named library library having following components (title, author-name, book-id, book-price).

```
#include<stdio.h>
struct library
{ char title[50];
  char author_name[50];
  float book_id;
  int book_price;
};
int main()
{ struct library b;
  printf("Enter book title:\n");
  scanf("%s", b.title);
  printf("Enter author name:\n");
  scanf("%s", b.author_name);
  printf("Enter book id:\n");
  scanf("%f", &b.book_id);
  printf("Enter book price:\n");
  scanf("%d", &b.book_price);
printf("The information of book shown below:\n");
printf("book title:%s\n", b.title);
printf("author name:%s\n", b.author_name);
printf("book id:%f\n", b.book_id);
printf("book price:%d\n", b.book_price);
return 0;
}
```

Page No.
Date / 20

* Local variable : A variable that are declared inside a function is called local variable. It can be used by those function in which it is defined. local variables are not known to outside the function.

Example :
```
#include<stdio.h>
int main()
{ int a;
  a=5;
}
```
-> local variable

* Global variable : A variable that are declared outside a function is called global variable. usually it declare top of the program. Global variable holds data for lifetime of your program and It can be used by inside the function as well as outside of the function.

Example :
```
#include<stdio.h>
int a=5;
int main(){
printf("%d", a);
}
return 0;
}
```

* Break statement with example :-

ans ⇒ 'break' is a keyword in C which is use to exit from loop.

Example :-

```
#include<stdio.h>
void main()
{ int i;
for(i=1;i<=10;i++)
{
if(i==6)
break;
printf("%d",i);
}
```

* Continue statement with example :-

ans ⇒ 'Continue' is a keyword in C which is used to ~~prevent~~ prevent the execution of a particular element or statement under the loop.

Example :-

```
#include<stdio.h>
void main()
{ int i;
for(i=1;i<=10;i++)
{ if(i==7)
continue;
printf("%d",i);
}
```

Page No.
Date / 20

* 4 stages of Compilation :-

(i) pre-processing
(ii) Compiling
(iii) Assembly
(iv) Linking

(i) pre-processing :- pre-processing is the first step in the compilation in c in this step all the statements which starting with # symbol is processed by pre-processor and ~~it converts~~ source code converts into an intermediate code with no # symbol.

(ii) Compiling :- Compiling step convert the intermediate (.i) file into an assembly file (.s).

(iii) Assembly :- Assembly step converts the assembly code into machine-understandable (binary code) code.

(iv) linking :- linking is a process of including the library files into our program.