Charlie

Start

Pre-requit of Programming in C

Charlie

Date

Page No.

* How to install code blocks on windows?

Google [ ]

mysirg.com > http://www.codeblocks.org/downloads

↑ quick link in footer

Note: Code Block IDE है Compiler नहीं है, Code Block के साथ-साथ Compiler भी install करना पड़ेगा जिसका नाम GNU GCC Compiler

So,

" Download the installer with GNU GCC compiler

eg.

codeblocks 16.01 mingw-setup.exe

(which includes MinGW & GNU GCC Compiler and GNU GDB debugger)

Code Block के website पे जाएँ :-

Download > download the binary release > windows ✓ > linux mac

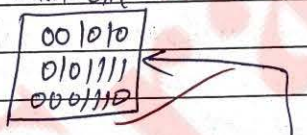
4th position वाला :-

codeblocks 16.01 mingw-setup.exe

* How to use code blocks for C languages

मान लो ये इस window के लिये software बनाना चाहते है

sum.exe

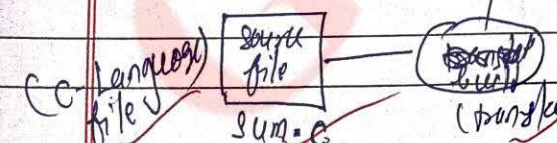


① sum.c] code blocks

② Build]

Run

• So, First save source file



Charlie

Date
Page No.

(L-1) 9:

start code block →
File > new > Empty file
↳ save as > sum.c

```
sum.c
main()
{
int a, b, c;
printf("enter two number");
scanf("%d %d", &a, &b);
c=a+b;
printf("sum of %d and %d is %d", a, b, c);
getch();
}
```

Build / Build and Run

Build करने पे दो file बनता है
sum.o (object file है)
sum.exe (executable file है) → यह software बन गया है)

यह .exe software को directly बना कर run भी कर सकते हैं।

Run

Enter two number
5
6
sum of 5 and 6 is 11

(L-1)g: Programming in C - Saurabh Shukla Sir (Charlie) Page No. Date

① Why C language is so important

- worth to know about C language
  - oracle is written C
  - core libraries of android are written in C
  - my SQL is written in C
  - almost every device driver is written in C
  - Unix operating system is developed in C
  - C is the world's most popular programming language
- For Student
  - C is important to build programming skills
  - C covers basic features of all programming language
  - Campus recruitment process

② History of C language

Martin Richards - Developer of BCPL, 1966
(Basic Combined Programming Language)

Ken thompson - Developer of B language
- 1969
- Also developer of Unix operating system

Dennis Ritchie - Developer of C language
- 1972
- Develope and co-founder of Unix operating system.

Charlie
Date
Page No.

start code block -
File > new > Empty file
↳ save as > sum.c

sum.c

```
main()
{
int a, b, c;
printf("enter two number");
scanf("%d %d", &a, &b);
c=a+b;
printf("sum of %d and %d is %d", a, b, c);
getch();
}
```

Build / Build and Run

build करते ही यो file बनता है
sum.o (object file है)
sum.exe (executable file है) → अब software बन गया है)

RUN

अब इस software को directly बना कर run कर सकते हैं।

enter two number
5
6
sum of 5 and 6 is 11

(L-1)g: Programming in C - Saurabh

① Why C language is so important

- worth to know about C language
  - oracle is written C
  - core libraries of android are written in C
  - my SQL is written in C
  - Almost every device driver is written in C
  - Unix operating system is developed in C
  - C is the world's most popular programming language.
- For Student
  - C is important to build programming skills
  - C covers basic features of all programming language
  - Campus recruitment process

② History of C language

martin Richards - Developer of BCPL, 1966
(Basic Combined Programming Language)

Ken thompson - Developer of B language
- 1969
- Also developer of Unix operating system

Dennis Ritchie - Developer of C language
- 1972
- developed and co-founder of Unix operating system.

L-1(2) Introduction to C part 2 hindi

3.) What is Computer?
- Computer is an electronic device that takes input, process it and gives output
- There is nothing like 0 and 1 in computer.
- There is no physical significance of 0 and 1.
- Any information can be encoded as a sequence of 0 and 1.

4.) How 0 and 1 get stored in memory.

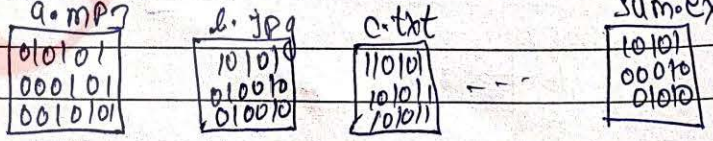
Ram - Capacitor < charge - 1 (hold) / non-charge - 0 (hold) } (charge so volatile)

magnetic tap - non-volatile

Processor memory - volatile

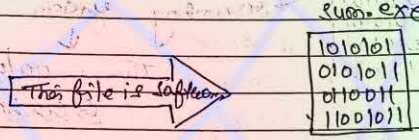
5.) What is hardware?
- Hardware is a comprehensive term for all of the physical parts of a computer, as distinguished from the data it contains or operates on, and the software that provides instructions for the hardware to accomplish tasks.
- Hardware is anything which is tangible (touchable, Physical)

6.) What is a File?
"File is a data bundle"
file name ↘ ↙ file extension



Charlie
Date
Page No.

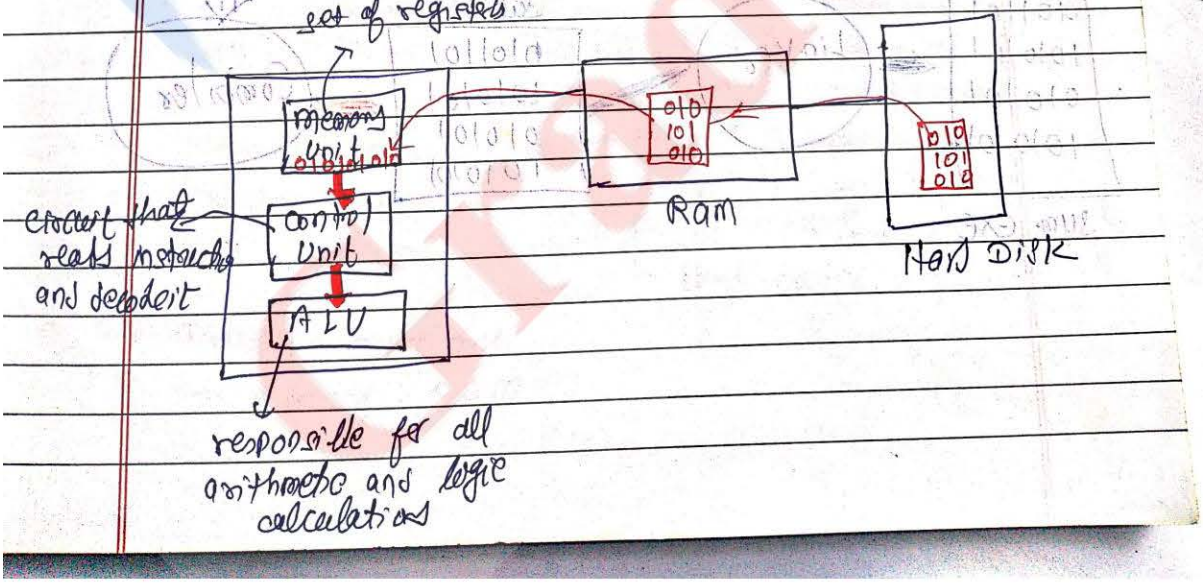
7.) what is software?
- Application software
- System software

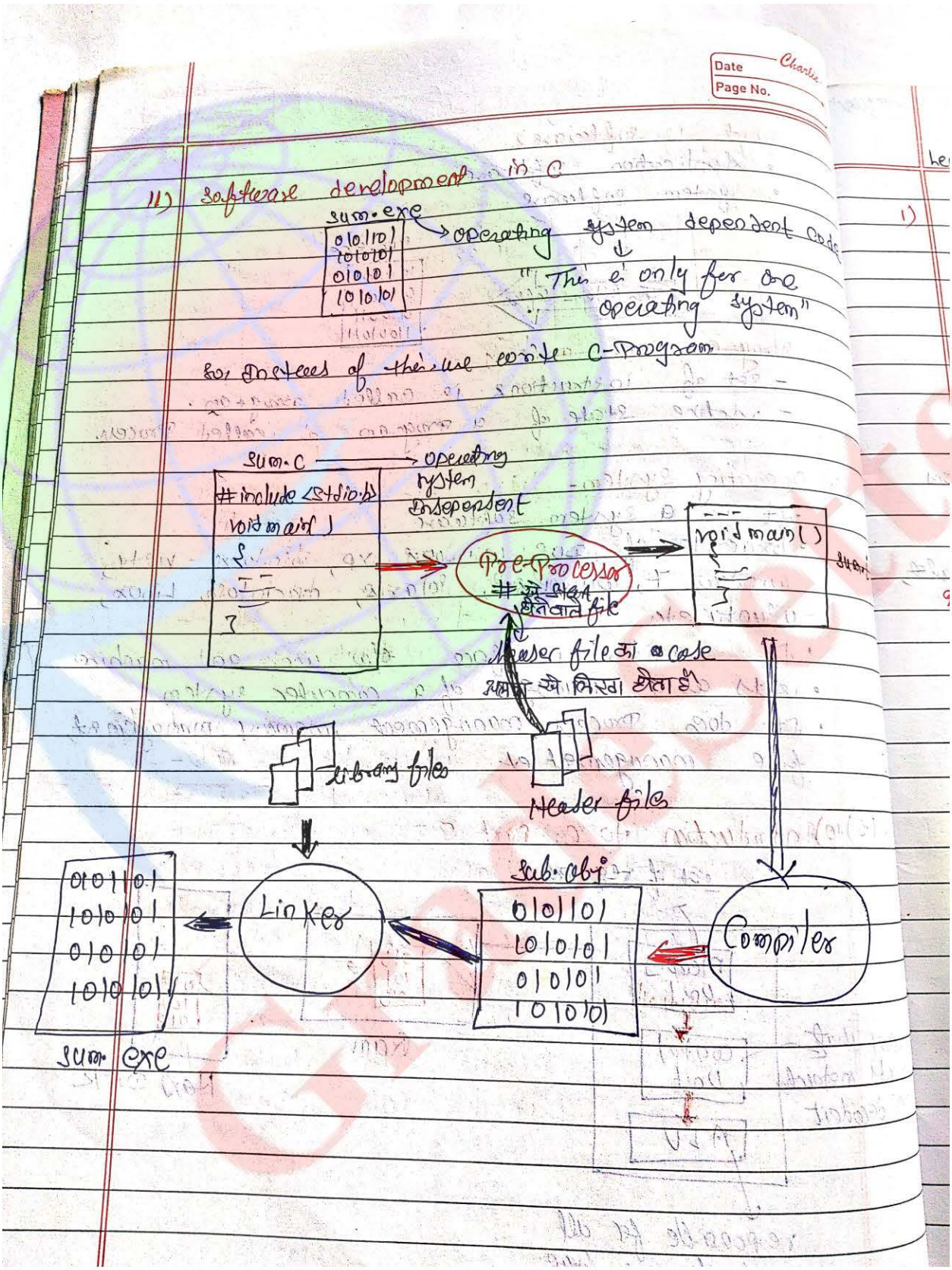


8.) Program and Process
- set of instructions is called program.
- Active state of a program is called Process.

9.) Operating System-
- It is a system software
- Example of DOS, windows xp, windows vista, windows 7, windows 8, solaris, macintosh, Linux, ubuntu etc.
- It provide interface b/w user and machine
- Acts as a manager of a computer system
- It does Process management, memory management, file management etc.

1-1(c) 10) Introduction to C Part 1





Lecture 2 - Identifier's in C Part-1

1) Begin learning C

A to Z a to z 0 to 9 (f: " / \ ? > < + = * & % !	1. Constant 2. variable 3. Keywords	1.) Data type declaration instruction 2.) Input / output Instruction 3.) Arithmetic Instruction 4.) Control Instruction
	Identifier	Instruction

Identifier → "smallest identifying unit in the Program"

Instruction ⇒ Program

2.) Constant

- Any information is constant
- Data = Information = Constant

Type of Constants:

(i) Primary constants
- Integer eg. -55 25 0 (point not allowed)
- Real eg. 21.4 3.56 -0.6
- Character eg 'a' 'B' '+' '2' (only one होता चाहिये)

~~'-5'~~ ~~'sahil'~~ ~~'24'~~ → Not character (This is mainly string)

(ii) Secondary constants
- Array
- Strings
- Pointer
- Union
- Structure
- Enumerator

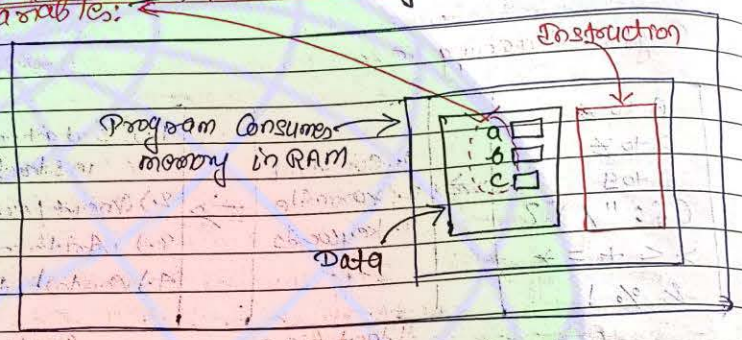
3.) Process memory

- memory : RAM

"C language is case sensitive"

Date
Page No.

4) Variables: → variable memory ki



"variables are the names of memory locations where we store data"

Rules:
- variable name is any combination of alphabets, digit and underscore.
- A valid variable name can not start with digit.

5) Keywords
- Predefined words
- Reserved words
- There are 32 keywords in C language

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

Lecture 3: Data type Declaration Instruction

1) what is Instruction
- Program statements are called Instructions
- Instructions are Commands
- Type of Instructions
  - * Data type Declaration Instruction
  - * Input Output Instruction
  - * Arithmetic Instruction
  - * Control Instruction

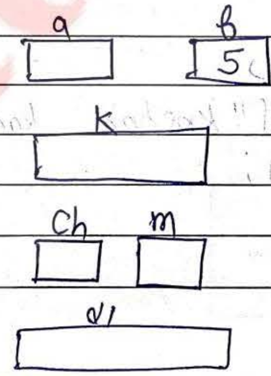
2) Data type
- int
- char
- float
- double
- void

3) Primitive Data type ⇒ Keyword भी है और data type भी है

auto, double, goto, signed, unsigned, break, void, int, char, float

4) Declaration statements

```
int a, b = 5;
float k;
char ch, m;
double d1;
```



Date Charlie
Page No.

Lecture 4: Input - Output Instruction

1.) - Keyboard is standard input device
- Monitor is standard output

2.) Printf ( )
- Printf( ) is not a keyword
- Printf( ) is a predefined function

- Two types of messages
  - Printing text as it is.
  - Printing value of expression or value of variable

Case sensitive
should be in small letter only

```
main( )
{
Printf ("Krishna Kanhaiya");
}
```

o/p :- Krishna Kanhaiya
(कृष्णा कन्हैया
लिखी जाएगी)

```
main( )
{
Printf (" Krishna Kanhaiya");
getch( ); ——> Character keyboard से लेता
}
```

o/p - Krishna Kanhaiya
(getch के कारण
जब input
लेने के लिए
screen रुकी रहेगी)

```
main( )
{
clrscr( )
Printf (" Krishna Kanhaiya");
getch( );
}
```

o/p - Krishna Kanhaiya
(screen clear
होकर
o/p आएगी)

Lecture 4(b): Input - Output Instruction

Charlie
Date
Page No.

(1)
```
main()
{
clrscr(); // to clear the screen
Printf("Krishna");
Printf("Kanhaiya");
getch();
}
```

o/p
Krishnakanhaiya

```
Printf("Krishna\n");
Printf("kanhaiya");
```

o/p
Krishna
Kanhaiya

```
Printf("Krishna\nkanhaiya");
```

o/p
Krishna
Kanhaiya

Escape sequences:-
\n
\t
\b
\\
\"
\r

(2) name को different Co-ordinate

```
main()
{
gotoxy(40,12);
getch();
}
```

o/p
-

```
main()
{
clrscr();
gotoxy(40,12);
Printf("Krishna");
getch();
}
```

o/p
-Krishna

Lecture(6): Input output Instruction

① main()
{
int a=4, b=5; - declaration statement
clrscr(); - action statement
}

declaration statement
action statement
के पहले ही होना चाहिये

② main()
{
int a=4, b=5;
clrscr();
Printf("a");
getch();
}

o/p
a

main()
{
int a=4, b=5;
clrscr();
Printf("%d", a);
getch();
}

o/p
4

Note: Format specifier

- %d
- %f
- %c
- %Lf

Printf("a=%d", a);

o/p
a=4

Printf("value of a is %d", a);

o/p
value of a is 4

Charlie
Date
Page No.

printf("value of a is %d and b is %d", a, b);

o/p

value of a is 4 and b is 5

printf("Sum of %d and %d is %d", a, b, a+b);

printf expression को भी solve कर सकता है।

o/p

Sum of 4 and 5 is 9

Lecture (d): Input output Instruction.

1.) scanf() - scanf is not a keyword
- scanf() is a predefined function
- scanf("format specifier", variable address);

scanf का काम है keyboard से data लेना और उस data को variable में store करना।

```
main()
{
int x;
clrscr();
scanf("%d", &x);
getch();
}
```

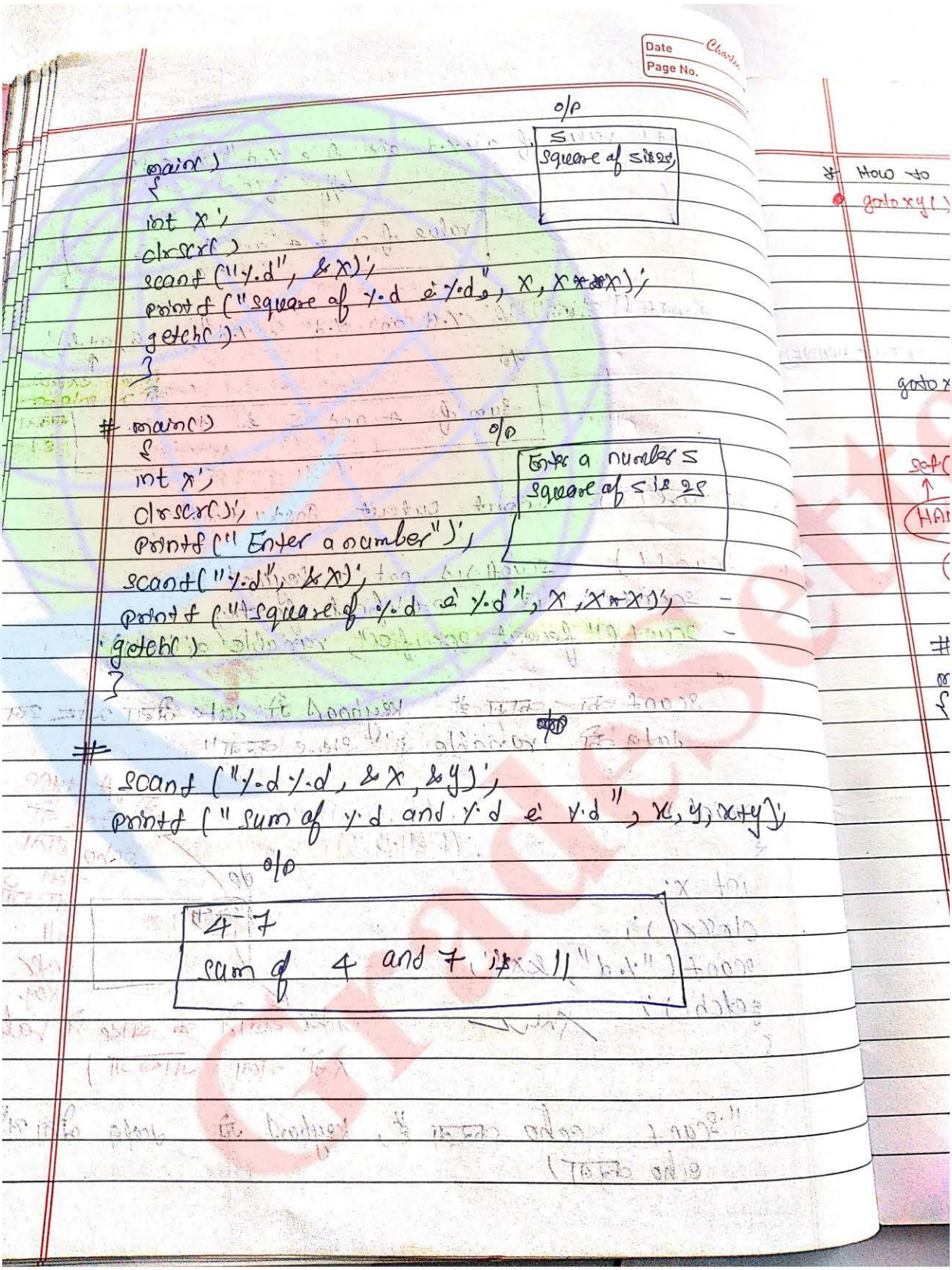
o/p

जो भी type करेंगे वो echo होता चला जाएगा। till enter key.

54

enter दबाने के बाद ये value x में चला जाएगा।

"scanf echo करता है, keyboard से data लेना और echo करना।



```
main()
{
int x;
clrscr();
scanf("%d", &x);
printf("square of %d is %d", x, x*x);
getch();
}
```

o/p

```
5
square of 5 is 25
```

```
# main()
{
int x;
clrscr();
printf("Enter a number");
scanf("%d", &x);
printf("square of %d is %d", x, x*x);
getch();
}
```

o/p

```
Enter a number 5
square of 5 is 25
```

```
#
scanf("%d%d", &x, &y);
printf("sum of %d and %d is %d", x, y, x+y);
```

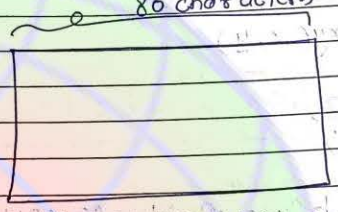
o/p

```
4 7
sum of 4 and 7 is 11
```

Charlie
Date
Page No.

* How to use gotoxy function in code Block

gotoxy() :



gotoxy(20, 8);

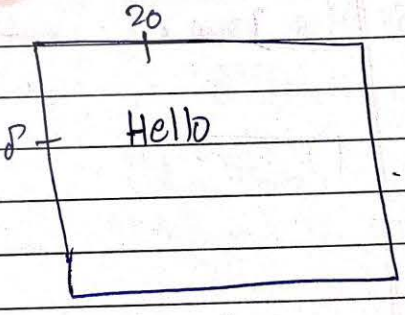
SetConsoleCursorPosition()

↑

HANDLE

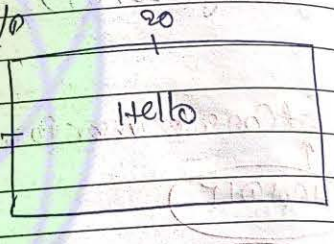
COORD ← यह एक structure है।

```
#include <window.h>
main()
{
  COORD c;
  c.X=20;
  c.Y=8;
  SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), c);
  printf("Hello");
  getch();
}
```



```
#include <windows.h>
void gotoxy(x,y)
{
 COORD c;
 c.X = x;
 c.Y = y;
 SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_H
}
main()
{
gotoxy(20,8)
printf("Hello");
getch()
}
```

o/p

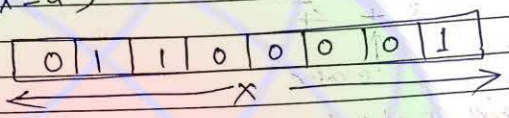



```
# main()
{
 int i;
 for(i=1; i<=10; i++)
   {
    gotoxy(10,i);
    printf("Hello");
   }
 getch();
}
```

Charlie
Date
Page No.

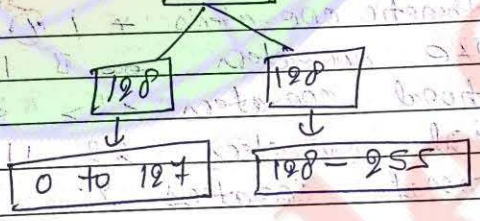
* What are ASCII codes?
American standard code for information interchange

char x = 'a';
1 byte memory



c);
- ASCII is a coding technique
eg.
'a' = 97
= 01100001 (8 bits)
'b' = 98
'' = 32
'0' = 48
'@' = 64

- ASCII code = 256



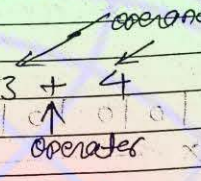
total 0 to 255
11111111

- ASCII is the numerical representation of a character such as 'a' or '@' or an action of some sort
- ASCII was actually designed for use with teletype.

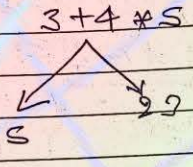
Date Charlie
Page No.

* Lecture 5 : Arithmetic Instruction :-

① operater



② Arithmetic Instruction :-
An instruction which is used to manipulate data using operaters, is known as arithmetic instruction.



Precedence rule : BODMAS (like)
Note - There is no BODMAS in C language.

operater groups :-
- Unary operater's + - ++ -- !
- Arithmetic operaters * / % (3 modules) + -
- Bitwise operaters & | ^
- Relational operaters < > <=
- Logical operaters && ||
- Conditional operaters ?:
- Assignment operaters =

UARBLCA

* Lecture 5(B) :- Unary operaters
+ eg +5
- eg -7
++ increment operater
--
size of( )

Charlie
Date
Page No.

```
main()
{
clrscr();
int x=3;
x++;          // x=x+1
printf("%d",x);
getch();
}
```

O/P 4

इसकी Priority सबसे कम होती है,

```
x++;  - Post increment
printf("%d",x);
++x;  - Pre increment
printf("%d",x);
```

O/P 4 5

```
x--;  - Post decrement
printf("%d",x);
--x;  - Pre-decrement
printf("%d",x);
```

O/P 2 1

```
main()
{
int x=3,y;
clrscr();
y=x++;
printf("%d%d",x,y);
getch()
}
```

यहाँ पर पहले assignment operator चलेगा।

O/P 4 3

Date
Page No.

```
main()
{
int x=3, y;
clrscr();
y=++x;
printf("%d %d", x, y);
getch();
}
```

o/p: 4 4

* Lecture (c) Arithmetic Instructions in C part 3

- size of (data type)
- sizeof (variable)
- size of (constant)

byte में answer देता की size बताती

```
main()
{
int x;
clrscr();
x = sizeof(float);
printf("%d", x);
getch()
}
```

→ 4 byte

o/p: 4

x = sizeof (double); o/p: 8

x = sizeof (char); o/p: 1

x = sizeof (int); o/p: 2

Date
Page No.

```
main()
{
int x,y;
float k;
double d1;
char ch;
clrscr()
x=sizeof( );
printf("%d", x);
getch();
}
```

[यहाँ पर variable लिख रहे हैं]

o/p

x = 2 (int)
y = 2 (int)
d1 = 8 (double)
ch = 1 (char)
k = 4 (float)

```
main()
{
int x,y,z;
clrscr()
x=sizeof(34);
y=sizeof(7.56);
z=sizeof('a');
printf("%d%d%d", x,y,z);
getch();
}
```

Real constant by default double होते हैं।

ASCII में int में change होजाता है।

o/p

2 8 2

* Modulas operators:-

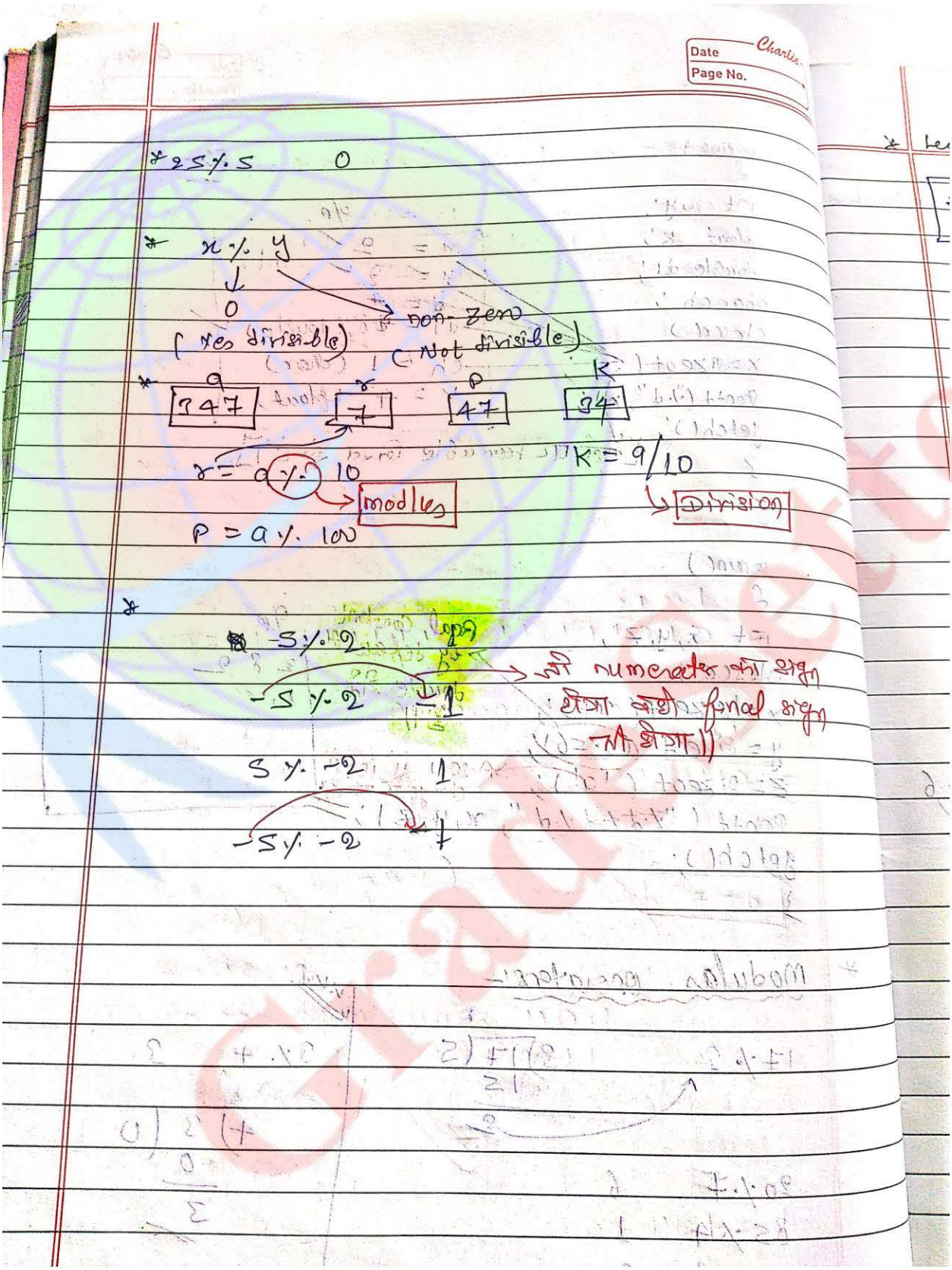
17 % 3 → 3)17(5, 15, 2

20 % 7 → 6
85 % 4 → 1
10 % 4 → 2

V.V.I

3 % 4 → 3

4)3(0, 0, 3



Charlie
No.

Charlie
Date
Page No.

* Lecture 5(d): Arithmetic Instructions:-

* / %
+ -

Associativity Rule → Left to Right

```
main()
{
int x;
x=3+4;
Printf("%d", x);
getch();
}
```

o/p: 7

3/4 → 0
3.0/4 → 0.75
3.0/4.0 → 0.75
3/4.0 → 0.75

जोड़ में एक भी real होगा तो answer real में आएगा।

3%4 → 3

* Lecture: Bitwise NOT operator:

~ → Bitwise operator
Unary operator

Turbo:
int = 2 byte
codeblocks:
int = 4 byte

```
int x;
x=~5;
Printf("%d", x);
```

5 = 00000000 00000101
~5 = 11111111 11111010

MSB: 0 ⇒ +ve
1 ⇒ -ve

Currently this is in 2's complement.

Page No.

Negative number stored in memory in 2's Comp.

-20 | 20 → (B1) —2's comp→ (B2)

20 ← (B2)

$x = b_1$ —2's→ $-x = b_2$

* $-x = b_2$ —2's→ $x = b_1$

* $5 = 00000000\ 00000101$

$\sim 5 = 11111111\ 11111010 = b_2$

1's comp: 00000000 00000101 +1

2's comp: 00000000 00000110 $= b_1 = 6$

(the)

So, $b_2 = -6$

Eg) $x = \sim 12$, (-13 (store हुआ है))

$12 = 00000000\ 00001100$

$\sim 12 = 11111111\ 11110011 = b_2$

(-ve no. है)

$-x = b_2$ —2's comp→ $b_1 = 00000000\ 00001101 = 13$

Charlie

Date
Page No.

Lecture 5 – Bitwise operator

* Bitwise operators –

Bitwise AND &
Bitwise OR |
Bitwise XOR ^
Bitwise NOT ~
Right shift >>
Left shift <<

a) & operator (AND operator)

0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1

eg. int x;
x = 23 & 56;

23 = 0000 0000 0001 0111
56 = 0000 0000 0011 1000
0000 0000 0001 0000 = 16

b) | OR operator

0 | 0 = 0
1 | 0 = 1
0 | 1 = 1
1 | 1 = 1

23 = 0000 0000 0001 0111
56 = 0000 0000 0011 1000
0000 0000 0011 1111 = 63

c) ^ (XOR operator)

0 ^ 0 = 0
1 ^ 0 = 1
0 ^ 1 = 1
1 ^ 1 = 0

23 = 0000 0000 0001 0111
56 = 0000 0000 0011 1000
0000 0000 0010 1111 = 47

d) Right shift

```
int x;
x = 56>>2;
```

56 = 0000 0000 0011 1000
14 = 0000 0000 0000 1110

e) Left shift

```
int x;
x = 56<<3;
```

56 = 0000 0000 0011 1000 000
448 = 0000 0001 1100 0000

* Lecture 5: Part 7: Logical operators
Not ! (higher priority)
AND &&
OR ||

* ! operator:-
- It is also operator
- Priority level is same as of many operators
- It reads the truth value of statement
!T = F , !F = T

```
main()
{
int y, x=5;
clrscr();
y = x > 4;
printf("%d", y);
getch();
}
```

o/p → true → 1

```
y = !x > 4
```

o/p → false → 0

&& :-

```
y = x > 4 && x < 10;
```

o/p → 1

Statement 1	&&	Statement 2	Result
F	X		F
T	F		F
T	T		T

```
y = x < 4 && x < 10;   —o/p
```

o/p → 0

|| :-

Statement 1	||	Statement 2	Result
F	F		F
F	T		T
T	X		T

```
y = x < 4 || x < 10;   o/p
```

o/p → 4 → true

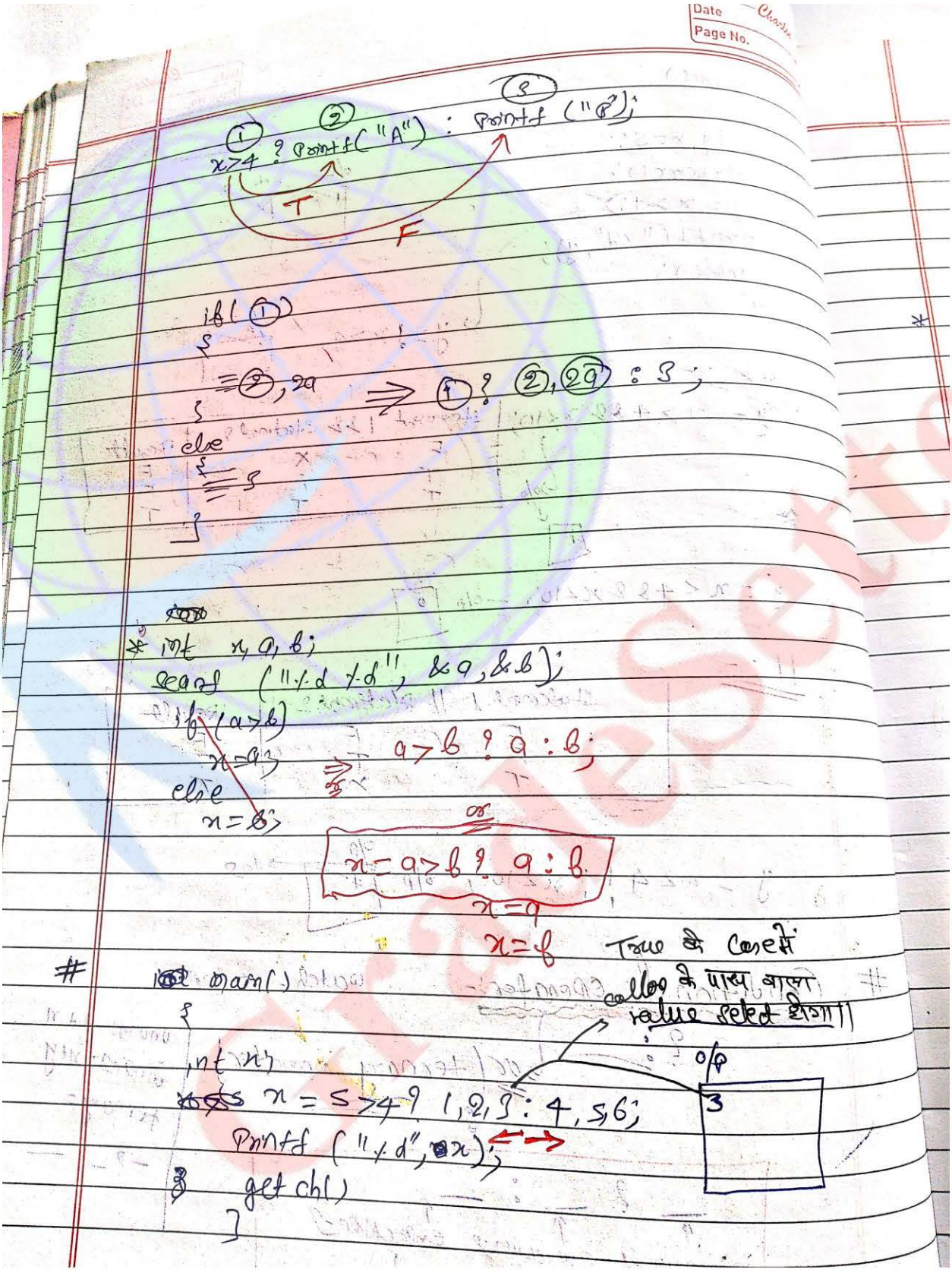
Conditional operater :- watch :- version

?:

or (ternary operater)

unary: ++x
binary: x+y
ternary: -?=:-

```
 ?   :
 ↑   ↑   ↑
expression-1  expression-2  expression-3
```



Charlie
Date
Page No.

```
x = 5<4 ? 1,2,3 : 4,5,6
```

F

o/p
4

*
```
int fun()
{
return (5>4 ? 5 : 4);
}
main()
{
int x;
x=fun();
printf("%d", x);
}
```

o/p
5

Note

~~Wrong~~ 5>4? return (5): return (4); — The is wrong statement
X

```
main()
{
int a, b;
printf("Enter two numbers");
scanf("%d %d", &a, &b);
printf("Greater number is %d", a>b? a: b);
getch();
```

Priority:-

Unary
Arithmetic
Bitwise & | ^ ~ >> <<
Relational >< >= <=
logical && ||
conditional ? :
Assignment

* (a > b ? x = a : x) = b;

① ② ③

* main()
{
int x, a = 4, b = 5;
a > b ? x = a : (x = b);
Printf("%d", x);
getch()
}

* Lecture 6: Decision Control in C

main()
{
line 1;
line 2;
line 3;
line 4;
line 5;
line 6;
}

Charlie
Date
Page No.

Decision / control / Instruction:

* Control Instruction:-
  - Decision Control instruction
  - Iterative Control instruction
  - Switch Case Control instruction
  - Goto Control instruction

a) Decision control instruction
   i) If
   ii) if-else
   iii) conditional Operator (? :)

i) If

```
main()
{
clrscr();
Printf("Hello");
Printf("Bye");
}
```

o/p

HelloBye

ii)

```
main()
{
clrscr()
Printf("Hello");
If(3>4)
{
Printf("Bye");
}
getch()
}
```

o/p

Hello

"If के case में अगर true होगा तभी तो next parenthesis के अंदर वाला statement जाएगा। False होने पे control loop बाहर आ जाएगा।।

```
if(3<4)  ---> TRUE
{
printf("Bye")
}
```

o/p: Bye

```
* main()
{
  int x;
  clrscr();
  Printf("enter a number");
  scanf("%d", &x);
  if(x>0)
  {
   Printf("positive number");
  }
  if (x<=0)
  {
  Printf("non-positive number");
  }
  getch();
}
```

* Lecture 6: Part 3: ~ Decision control statements

```
if(condition
{
line 1;
line 2;
---
}
else
{
line1;
line2;
---
}
```

⇒ Condition ? statement : statement;

"Conditional statement is also used as relative assignment"

Charlie
Date
Page No.

```
main()
{
int x;
clrscr();
printf("enter a number");
scanf("%d", &x);
if(x>0)
printf("Postive");
else
printf("non-positive");
getch()
}
```

⇒ x>0 ? printf("Postive") : printf("non-Postive");

Selective assignment:

```
main()
{
int x, y, max;
clrscr();
printf("enter two numbers");
scanf("%d%d", &x, &y);
max = x>y ? x : y;
printf("greater no. is %d", max);
getch()
}
```

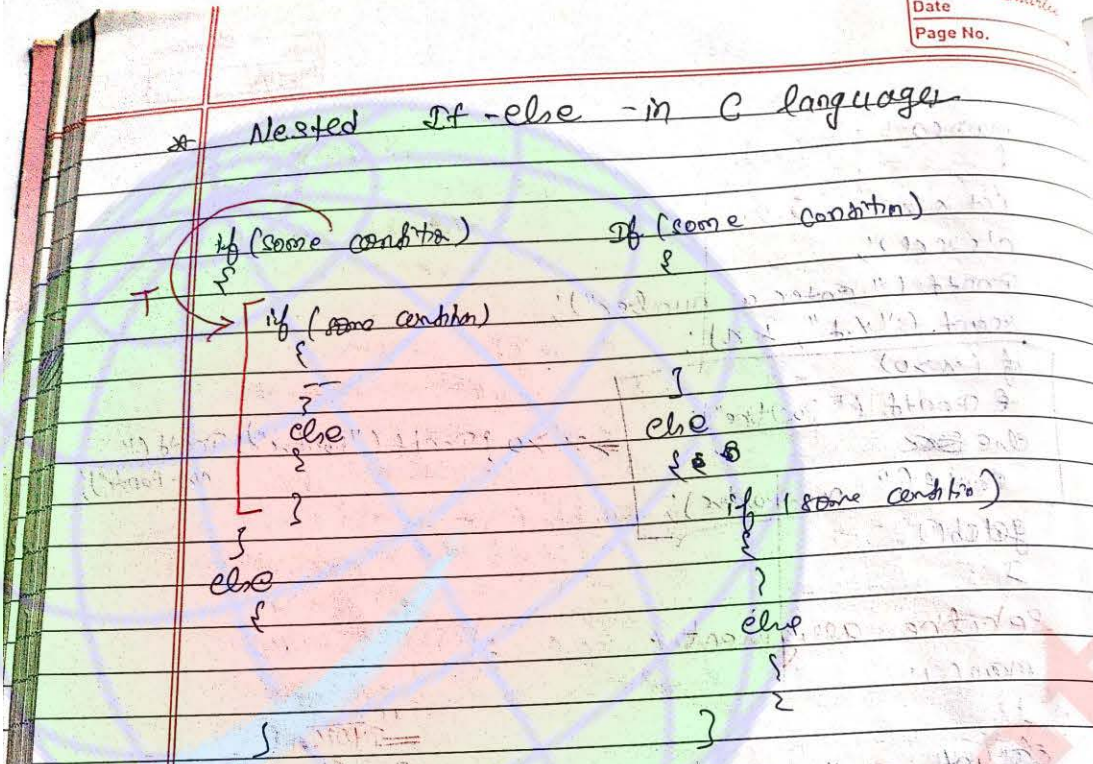
"इसमें variable बनाते हैं तो जो maximum value store होगी वो assign हो जाएगी"

or

```
printf("greater no. is %d", x>y ? x : y);
```

Date
Page No.

Nested If-else in C language




```
main()
{
int a, b, c;
Printf("enter three numbers");
Scanf("%d%d%d", &a, &b, &c);
if (a>=b && a>=c)
    Printf("%d", a);
if (b>=a && b>=c);
Printf("%d", b);
if (c>=a && c>=b);
    Printf("%d", c);
getch();
}
```

```
if (a>=b && a>=c)
    printf("%d", a);
else
{
    if (b>=a && b>=c);
        Printf("%d", b);
    else
        printf("%d", c);
}
getch();
}
```

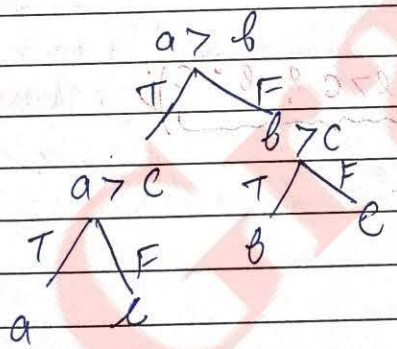
or

```
if (a>=b && a>c);
printf("%d", a);
else
{
    if (b>c)
        printf("%d", b);
    else
        Printf("%d", c)
}
```

o/p

```
10 20 70
30
```

or



```
if (a>b)
{
  if (a>c)
    printf ("%d", a);
  else
    printf ("%d", c);
}
else
{
  if (b>c)
    printf ("%d", b);
  else
    printf ("%d", c);
}
getch();
}
```

conditional operat

```
a>c ? printf("%d", a) :
      printf("%d", c);
```

conditional one

```
b>c ? printf("%d", b);
      printf("%d", c);
```

```
a>b ? a>c ? printf("%d", a) : printf("%d", c) : b>c ?
printf("%d", b) : printf("%d", c);
```

or

```
printf("%d", a>b ? a>c ? a : c : b>c ? b : c);
```

Charlie
Date
Page No.

If - else ladder in C :

```
if( )
    statement;
else if( )
    statement;
else if( )
    statement;
else if( )
    statement;
else
    statement;
```

```
main( )
{
    int marks;
    printf("enter number");
    scanf("%d", &marks);
    if (marks > 90)
        printf("Grade-A");
    else if (marks > 80)
        printf("Grade-B");
    else if (marks > 70);
        printf("Grade-C");
    else
        printf("Grade-D");
    getch( );
```

```
if( )
{
}
else
{
    if( )
    {
    }
    else
    {
        if( )
        {
        }
        else
        {
        }
    }
}
```

o/p

```
enter number
74
Grade-C
```

Lecture 7 : loop in C part - 1

*29.) 6.) Iterative Control Instruction

- → while
- → do while
- → for

means repetitive

"इसमें Control अपने आप आने वाले स्थान से चला जाता है।। मतलब इसका मैन सब गए हैं कि यह Condition करके अपने initial में automatically चला जाएगा जब तक Condition true होगा।।"

```
main( )
{
clrscr( );
Printf("krish");
getch( );
}
```

o/p

Krish

While :-

```
main( )
{
clrscr( );
while(Condition)
{
Printf("krish");
}
getch( )
}
```

Control फिर से while के पास ही जाएगा।

"while के case में Control बार-बार अपने पहले पहले वाले while पर ही आएगा।।

```
while("1<5")
{
Printf("k");
}
```

o/p

KKK —
Infinite time

```
main( )
{
int i=1;    // initialization
clrscr( );
while (i<=5)    // condition
{
Printf("K");
i++;    // flow
getch( );
}
```

o/p

KKKKK

Lecture 7 : loop in C part -1

Date
Page No.

*29.) 6.) Iterative Control Instruction

- while
- do while
- for

means repetitive

इसमें Control अपने आप अपने पहले वाले स्थान पे वापस आता है।। "मतलब इसका मतलब है कि यह Condition चेक करके अपनी initial पे automatically चला जाएगा जब तक Condition true होगा।।

```
main( )
{
clrscr();
Printf("krish");
getch();
}
```

o/p: Krish

While :-

```
main( )
{
clrscr( );
while(Condition)
{
Printf("krish");
}
getch()
}
```

Control फिर से while के पास ही आएगा।

"while के case में Control बार-बार अपने पहले पहले वाले while पर ही आएगा।

```
while(1<5)
{
Printf("K");
}
```

o/p: K K K — Infinite time

```
main()
{
int i=1;    // initialization
clrscr();
while(i<=5)    // condition
{
Printf("K");
i++;    // flow
getch();
}
```

o/p: KKKKK

Charlie
Date
Page No.

Lecture 4: loop in C Part-2

While

```
main()
{
int i=1;
while(i<=5)
{
Printf("krish");
i++;
}
getch();
}
```

6 comparision

total 6 comparision

- entry level comparision

o/p

krish krish krish krish krish

(5 times)

do-while

```
main()
{
int i=1;
do
{
Printf("krish");
i++;
}while(i<=5);
getch();
}
```

5 comparision

total 5 comparision

- exit level comparision

o/p

krish krish krish krish krish

(5 times)

For

```
main()
{
int i;
for(i=0; i<=5; i++);
{
Printf("krish");
}
getch();
}
```

total 6 comparision

- entry level comparision same as while

o/p

krish krish krish krish krish

(5 times)

Lecture 4: Part 2

Break:-

- The keyword "break" can be used in loop body or in switch body
- The purpose of break is to terminate loop's execution immediately as it encounters.

Date
Page No.

```
main()
{
int i=1, x;
while (i<=5)
 {
   printf("enter a number");
   scanf("%d", &x);
   if (x>0)
     break;
   i++;
 }
i==6 ? printf("ends normally") : printf("applies break");
getch();
}
```

* Lecture 8:- Switch Control in C:-

```
switch(expression)
{
case constant: code;
case constant: code;
case constant: code;
case constant: code;
default code;
}
```

should be unique (integer/character constant is allowed)

```
switch (expression)
 {
   case constant: code; break;
   case constant: code; break;
   case constant: code; break;
   case constant: code; break;
   default: code;
 }
```

Charlie
Date
Page No.

* Write a menu driven program with the following option.
1.) addition
2.) odd-even
3.) printing first n natural numbers

```
main()
{
int choice, a, b, s;
while(1) {
clrscr();
Printf("\n1. addition");
Printf("\n2. odd-even");
Printf("\n3. printing n numbers");

Printf("\n\n enter your choice");
scanf("%d", &choice);
switch (choice)
{
case 1:
        Printf("enter two numbers");
        scanf("%d %d", &a, &b);
        s=a+b;
        Printf("\n sum is %d", s);
        break;
case 2:
        Printf("enter a number");
        scanf("%d", a);
        if(a%2==0)
            Printf("even number");
```

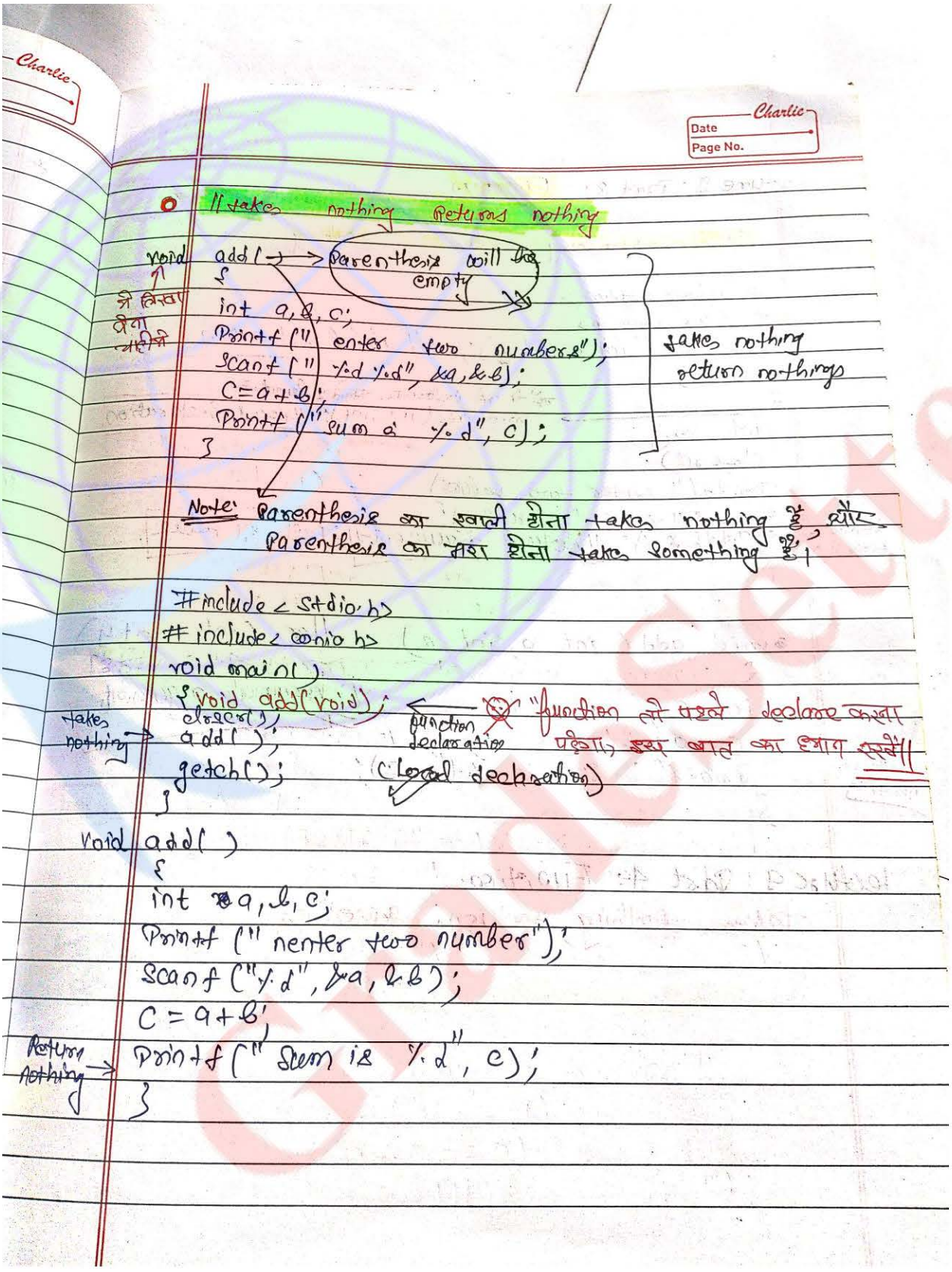
```
else
    Printf("odd");
break;
case 3:
Printf("enter a number");
scanf("%d", &a);
for(b=1; b<=a; b++)
    Printf("%d", b);
break;
case 4: exit(0);
default:
    Printf("invalid choice");
getch();
}
}
```

सतत्त program ही end हो जाएगा

Program को normally end कर रहे हैं

* Lecture 9: Functions in C

- Ways to define a function -
  - takes nothing, Return nothing
  - takes something, Return nothing
  - takes nothing, return something
  - takes something, Return something



// take nothing Returns nothing

void add() → Parenthesis will be empty

{
int a, b, c;
Printf (" enter two numbers");
Scanf (" %d %d", &a, &b);
c = a + b;
Printf (" sum is %d", c);
}

take nothing return nothings

Note: Parenthesis का खाली होना take nothing है, और Parenthesis का भरा होना take something है।

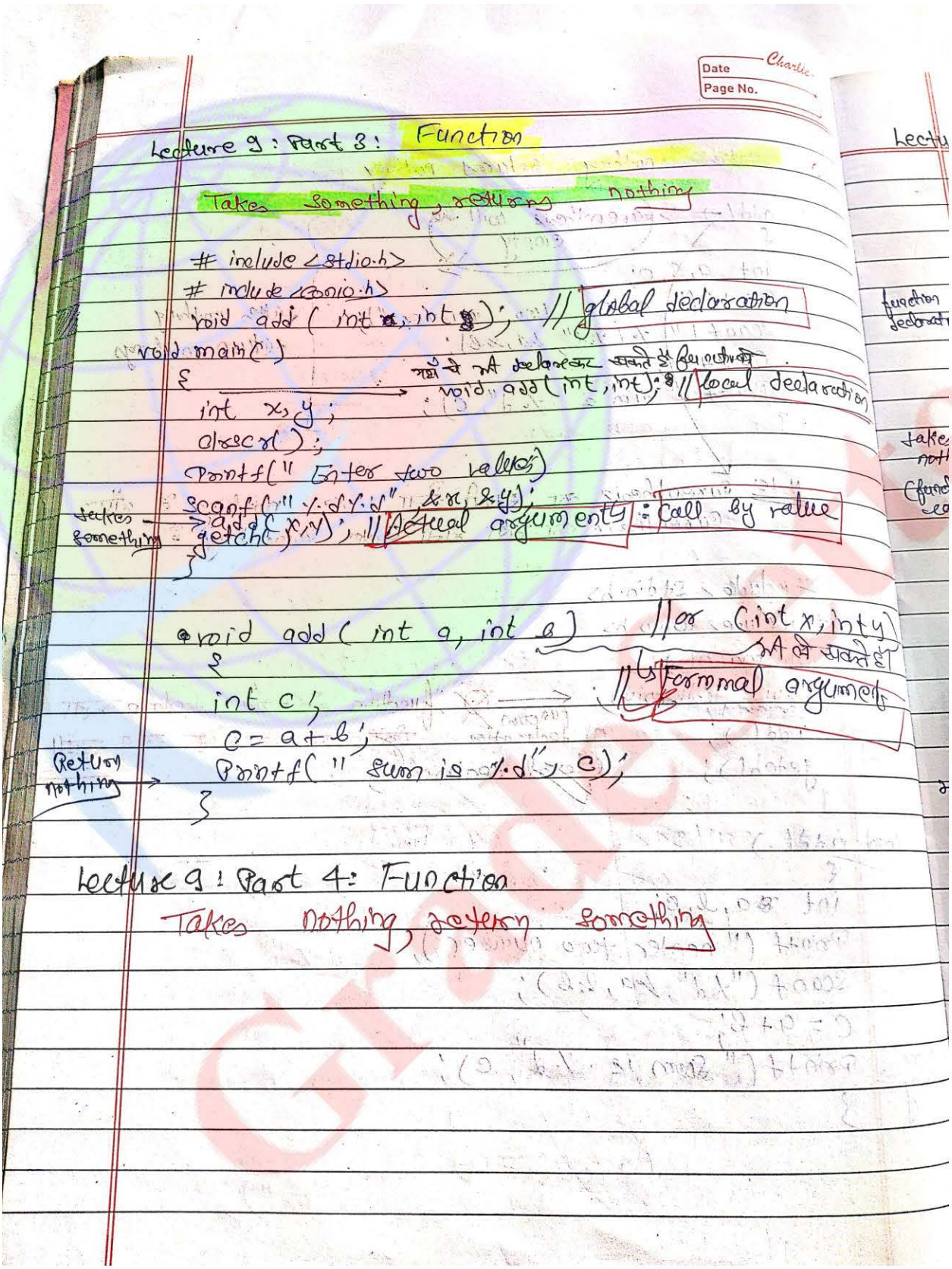
```
#include <stdio.h>
#include <conio.h>
void main()
{ void add(void);
clrscr();
add();
getch();
}
void add()
{
int a, b, c;
Printf (" nenter two number");
Scanf ("%d", &a, &b);
c = a + b;
Printf (" Sum is %d", c);
}
```

take nothing →

Return nothing →

function declaration (Local declaration)

"function से पहले declare करना पड़ेगा, इस बात का ध्यान रखें।"



Lecture 9 : Part 3 : Function

Takes something, returns nothing

```
#include <stdio.h>
#include <conio.h>
void add(int, int);          // global declaration
void main()
{
    void add(int, int);      // local declaration
    int x, y;
    clrscr();
    printf("Enter two values");
    scanf("%d %d", &x, &y);
    add(x, y);               // Actual arguments : call by value
    getch();
}

void add(int a, int b)       // or (int x, int y)
{                            // Formal arguments
    int c;
    c = a + b;
    printf("sum is %d", c);
}
```

Takes something → Return nothing

Lecture 9 : Part 4 : Function

Takes nothing, return something

Charlie
Date
Page No.

Lecture 9: Part 4: Functions in C

// Takes nothing return something

```
#include <stdio.h>
#include <conio.h>
int add(void);                    // function declaration
void main()
{
int s;
clrscr();
s=add();                          // takes nothing (function call)
Printf(" sum = %d", s)
getch();
}

int add()                         // control moves back
{
int x, y, c;
Printf(" enter two number");
scanf(" %d %d", &x, &y);
c = x+y;
return(c);                        // return something
                                  // or return(x+y);
Printf("hello");  <- Unreachable code
}
```

/* Takes something, Return something */

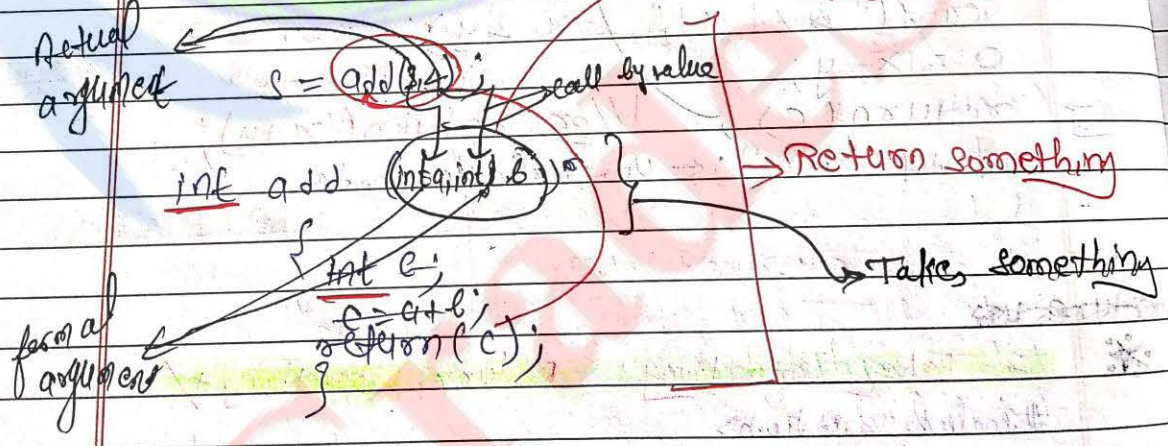
```
#include <stdio.h>
#include <conio.h>
int add(int, int);
main()
{
```

Charlie
Date
Page No.

```
int s,x,y;
clrscr();
Printf("enter the value");
scanf("%d %d", &x, &y);
s = add(x,y);
Printf("Sum is %d", s);
getch();
}

int add(int a, int b)
{
  int c;
  c = a+b;
  return(c)
}
```

takes something

Lecture 9: part 5: Function



/* takes something, returns something */

int add(int, int);

main()
{

इसमें ये function declaration में comma लगता है।

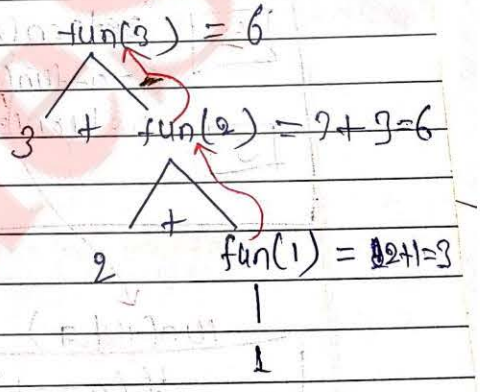
Charlie
Date
Page No.

```
int x, y, s;
printf("enter the value");
scanf("%d%d", &s);
s = add(x, y);
printf("sum is %d", s);
getch()
}

int add(int x, int y)
{
int c;
c = x+y;
return(c);
}
```

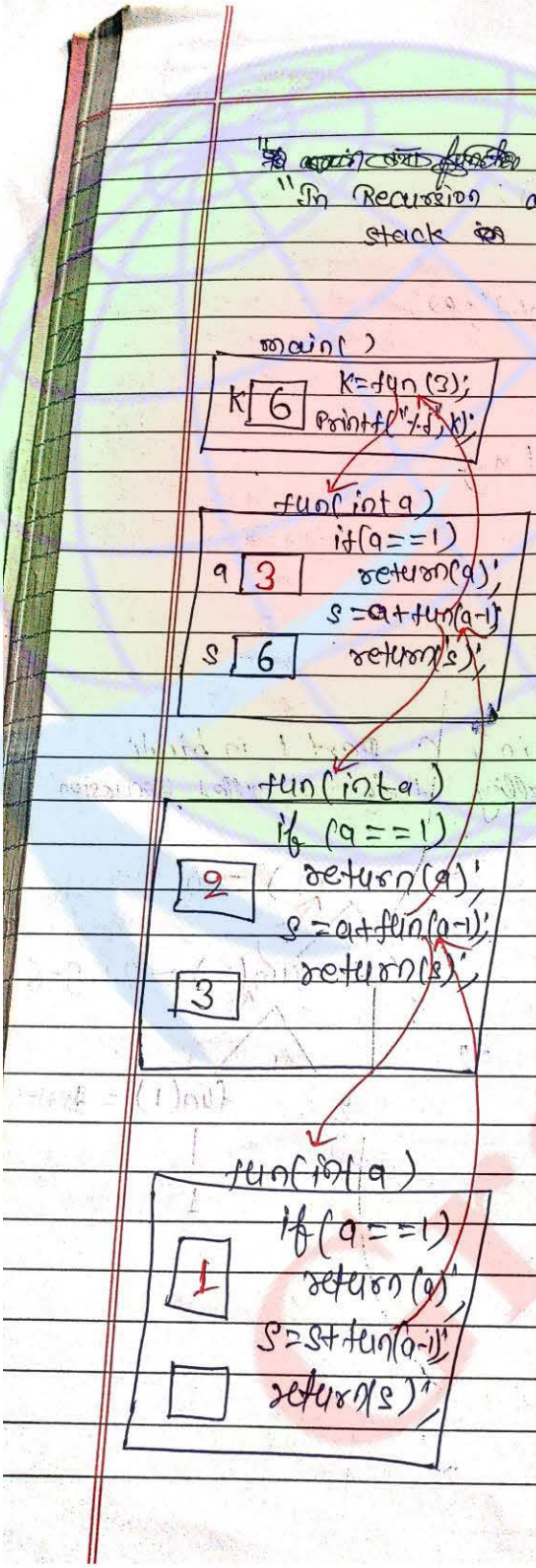
★ Lecture 10 : Recursion in C Part 1 in hindi

↳ Function calling itself is called Recursion

```
main()
{
int k;
k = fun(3);
printf("%d", k);
}

int fun(int a)
{
int s;
if (a==1)
return(a);
s = a + fun(a-1);
return(s);
}
```



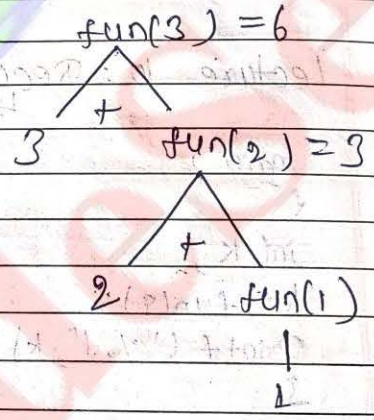
"In Recursion always activation record in stack is created."



sum(3) = 3 + sum(2)

sum(2) = 2 + sum(1)

sum(1) = 1



"In Recursion, each time the function call itself with a slightly simpler version of the original problem"

IT India : Recursion and stack

How Recursion works?

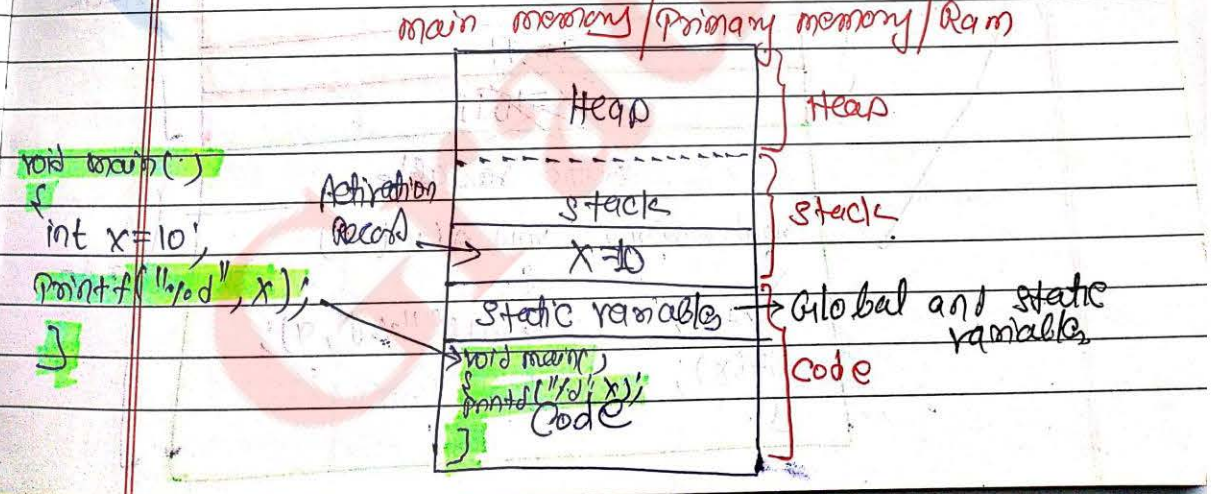
- main memory
- function call
- Nested function calls
- Recursion example 1
- Recursion example 2

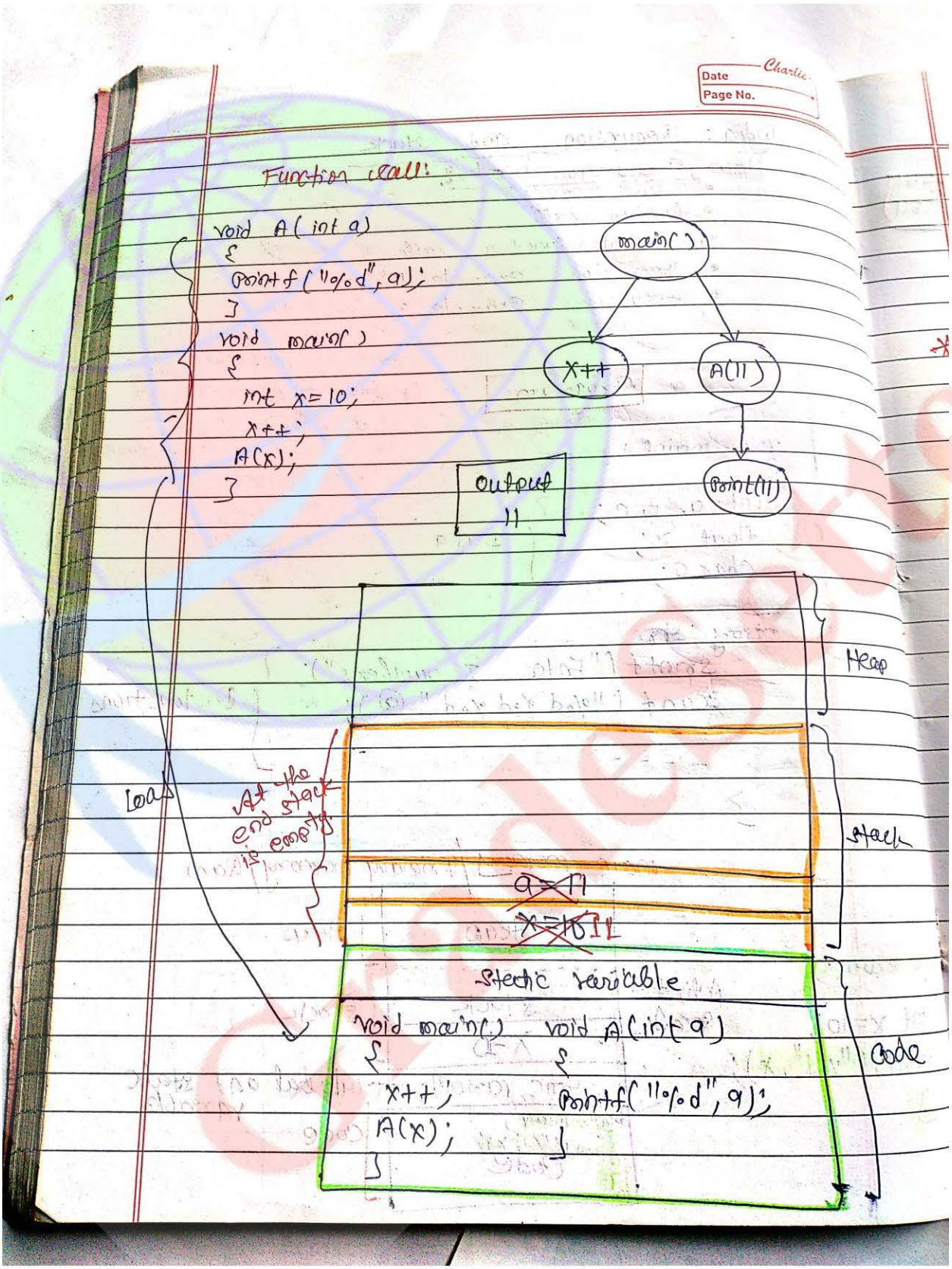
Program

```
void main()
{
    int a, b, c;
    float x;
    char c;
    printf("Enter 3 numbers");
    scanf("%d %d %d", &a);
    ----
    ----
}
```

Data

Instructions

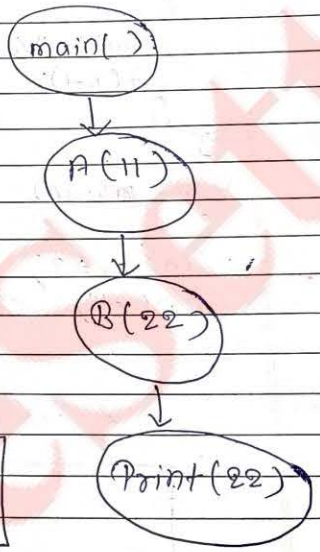




Note:
- When a function is called it's activation record is created
- When a function is terminated it's activation record is deleated

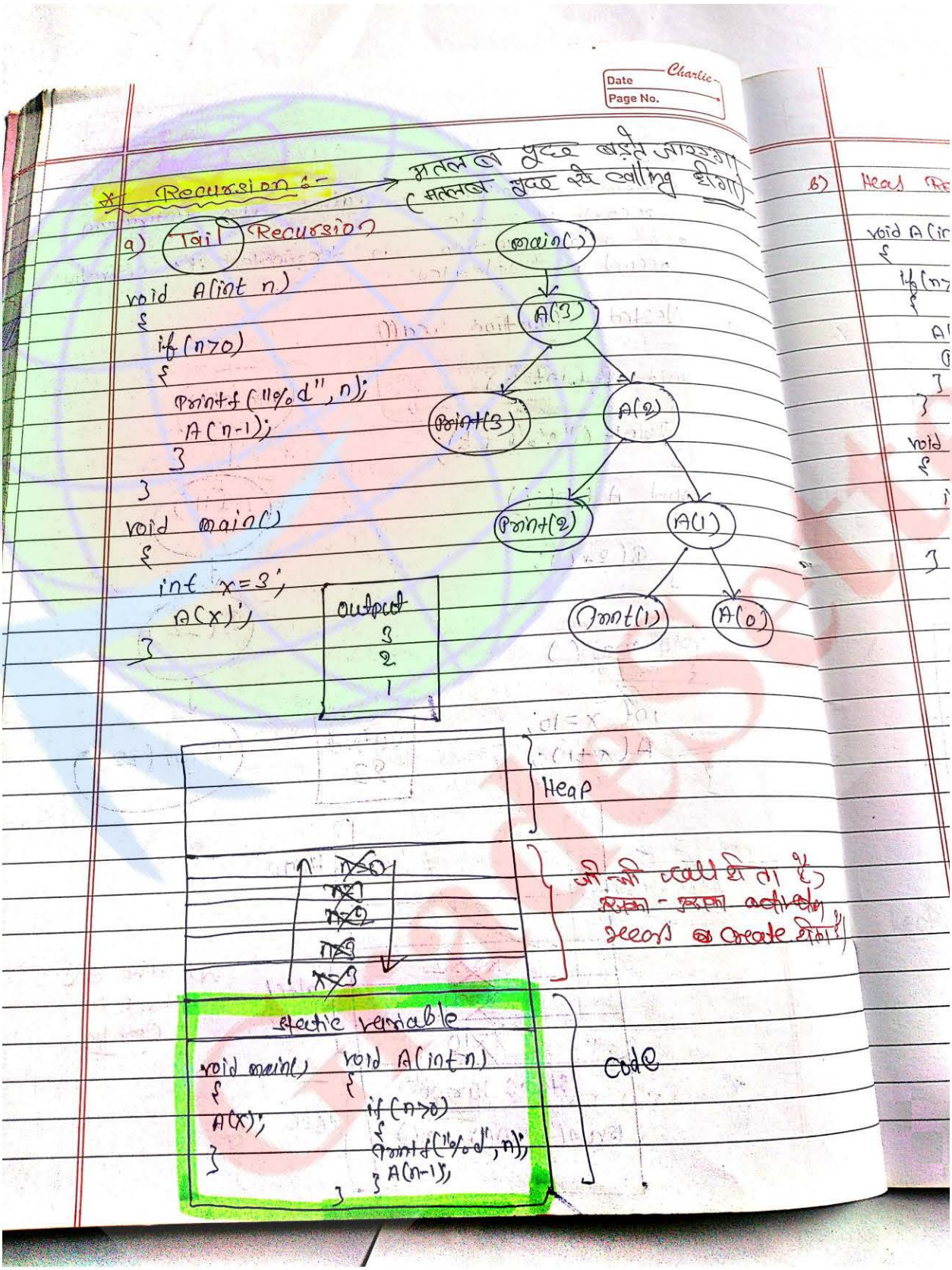
* Nested function call

```
void B (int k)
{
  Printf ("%d", k);
}
void A (int i)
{
  B(2*i);
}
void main()
{
  int x=10;
  A(x+1);
}
```



Output
22

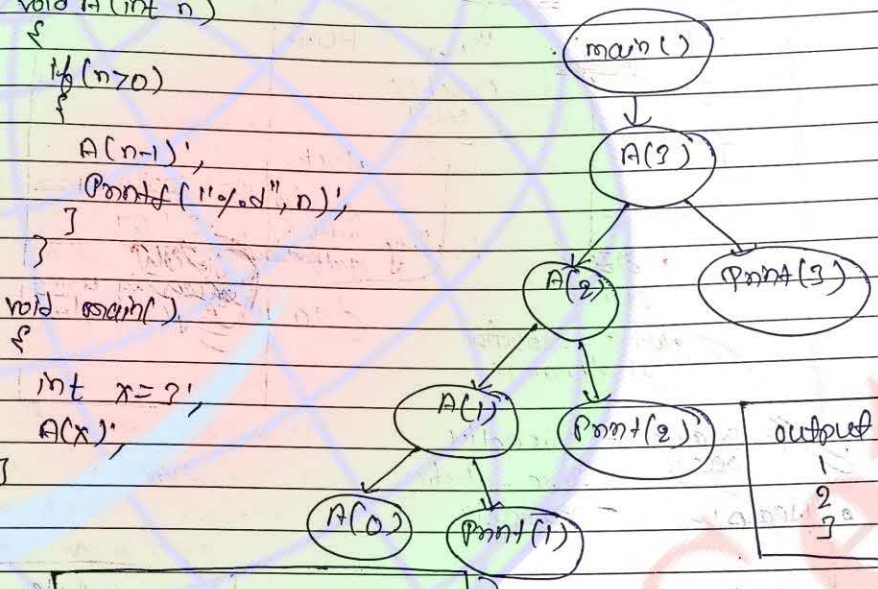
At the end stack is empty

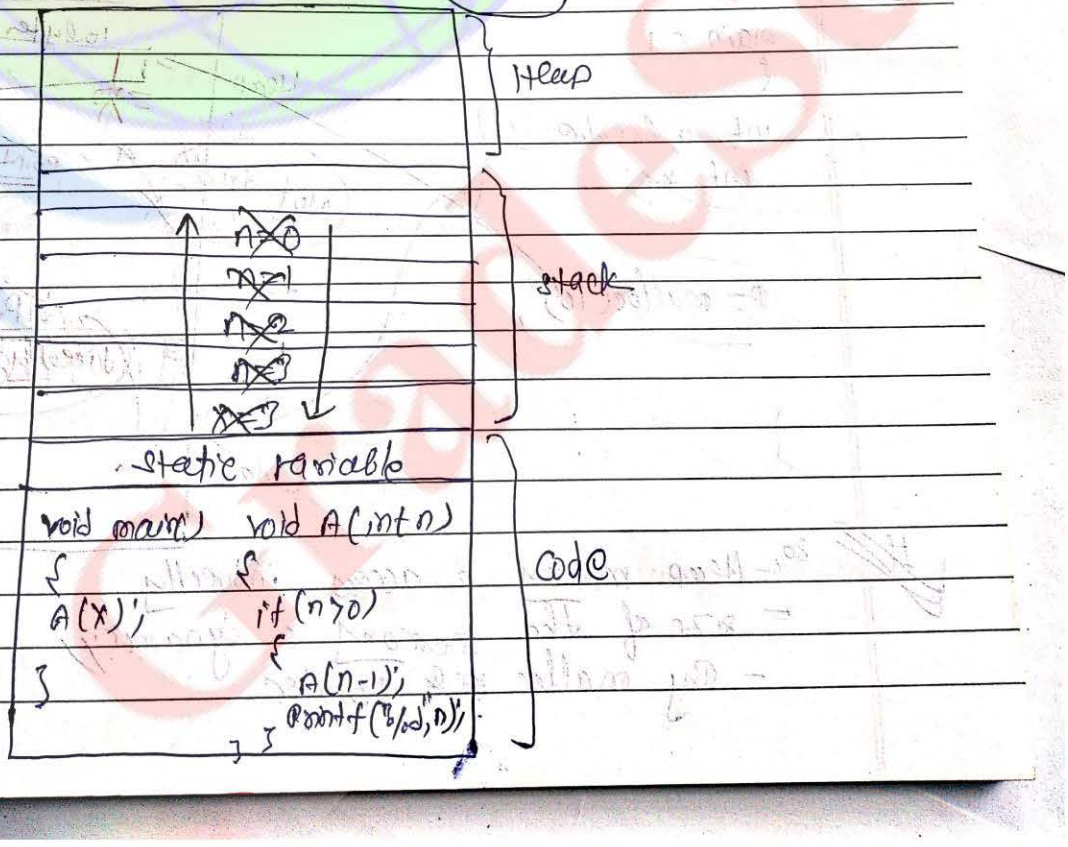


Charlie
Date
Page No.

6) Head Recursion:

```
void A(int n)
{
  if(n>0)
  {
    A(n-1);
    Printf("%d", n);
  }
}
void main()
{
  int x=3;
  A(x);
}
```





Date
Page No.

Lecture 11

IT Indiq : stack Vs Heap

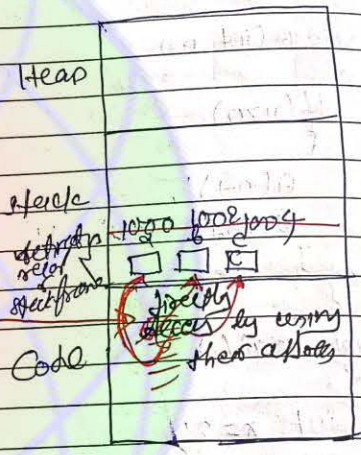
- stack:

```
main()
{
  int a,b,c;
  ---
  ---
}
```

Compile → m.c (machine code)

Loading

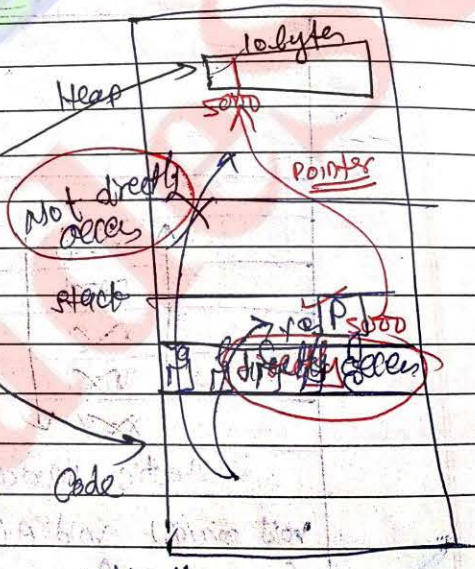
- Size is static
- while allocation is dynamic



stack - Directly access - size - static

- Heap:- - variables

```
main()
{
  int a,b,c,d,e;
  int *p;
  ---
  ---
  p = malloc(10);
  ---
  ---
}
```

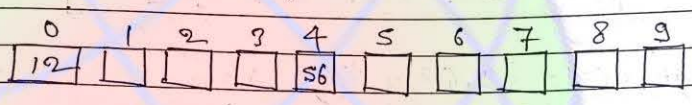


So, - Heap memory is access indirectly
- size of the memory is dynamic
- By malloc & pointer

Charlie
Date
Page No.

Lecture 11: Array in C part-1

- Array is a linear collection of similar elements
- Array is also known as subscript variable
- Array is a group of variable



a[0] = 12 ;
A[4] = 56 ;

int a[10]; → ये 10 variable को बिखा रहा है

```
main()
{
int a[10], i, sum=0;
float avg;
printf("Enter the number");
for(i=0; i<=9; i++)
scanf("%d", &a[i]);

for(i=0; i<=9; i++)
sum = sum + a[i]
avg = sum/10.0;
printf("average is %f", avg);
getch();
}
```

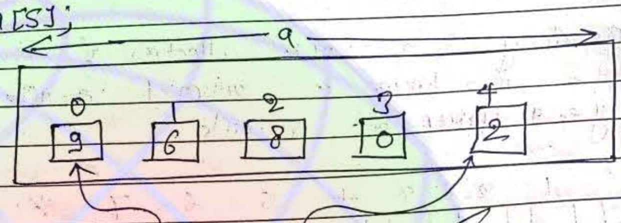
Lecture 11: Array in C Part 2

- int a[ ]; //error

  ↓ This is an error you have to always mention size

- int a[5]; ✓

- int a[5];



- int a[5] = {9,6,8,0,2};

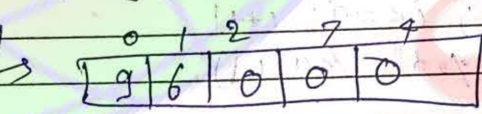
or

- int a[] = {9,6,8,0,2};

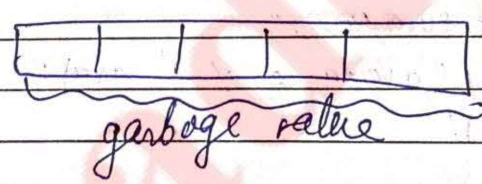
↓ Declaration के वक्त हम खाली छोड़ सकते हैं, अगर data भी भर रहे हो तो।

- int a[5] = {9,6,8,0,2,4,7}; //error

- int a[5] = {9,6};

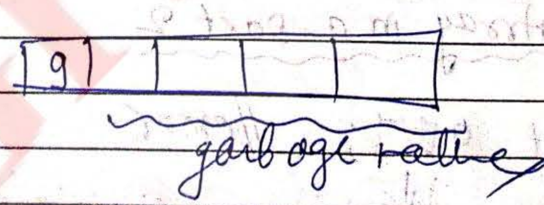


- int a[5];



If

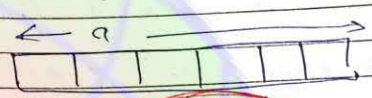
a[0]=9



Charlie
Date
Page No.

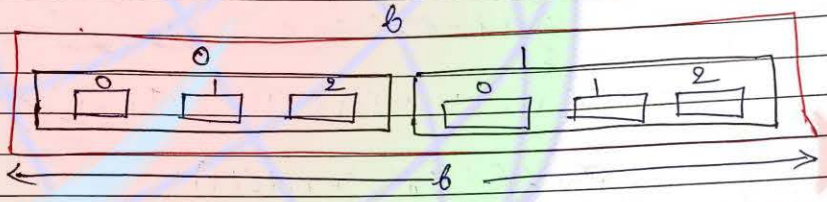
lecture 12: Two dimensional array in C

- int a[6];
  - one-dimensional array
  - (6 variable)



- int b[2][3];
  - Two dimensional array

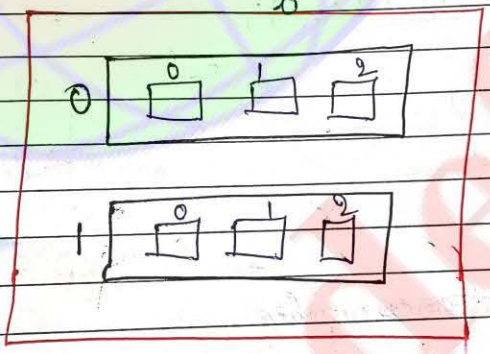
physically:



or

b

logically:




```
#include <stdio.h>
#include <conio.h>
main()
{
int A[3][3], B[3][3], C[3][3], i, j;
Printf (enter 9 numbers);
for (i=0; i<=2; i++)
    for (j=0; j<=2; j++)
```

```
scanf("%d", &A[i][j]);

Printf("enter the value");
for(i=0; i<=2; i++)
    for(j=0; j<=2; j++)
    scanf("%d", &B[i][j]);

for(i=0; i<=2; i++)
    for(j=0; j<=2; j++)
    {
C[i][j] = A[i][j] + B[i][j];

Printf("%d", C[i][j]);
    }
    Printf("\n");
}
getch()
{
```

output

enter a number

```
0 1
1 0
2 2
0 2
1 1
2 4
2
0
0
```

enter the value

```
2
2
2
4
5
1
0
0
```

```
2 2 4
+ 5 5
2 0 3
```

Charlie
Date
Page No.

Q2: When to use array in a program

When should we use an array

a) account no

age

roll no

marks

team-size

No logical connection

10

① int age, roll no, ... ✓ so the a' uses

or

② int a[10];

b) class 10 students

marks

= logical connection

i) int a student, marks

or

ii) int a[10]; ✓

c) class 10 students

int m[10];

Date
Page No.

Student number Students

class 0 1 2 3 4 5 6 7 8 9

d) class number

Real world में data two dimension में

so, int m[5][10]; ✓

class student no.

eg. class no. 2 के student 4 के marks

m[3][4]

d) int m[10][5][10];

institute no. class no. student number

Lecture 12: Strings

- sequence of characters terminated at null character
- ASCII code of null character is 0(zero)

```
main()
{
char s[10]={'S','A','V','R','A','R','H','\0'};
int i;
disco();
```

\0 Zero का ASCII code Zero

null character

0	1	2	3	4	5	6	7	8	9
S	A	V	R	R	A	R	H		

s

Charlie
Date
Page No.

```
for (i=0 ; i<=7 ; i++)
printf ("%c", s[i]);
getch();
}
```

output:
SAURABH _

or

```
for (i=0; s[i] != '\0'; i++)
      printf ("%c", s[i]);
```

or

```
printf ("%s", s); / printf ("%s", &s[0]);
```

↑ string

output: saurabh_

line printf cursor नीचे / same outputs

or

```
puts(s); / puts(&s[0]);
```

output
saurabh
_

or

```
char s[10] = "SAURABH";
```

use of string.

output
SAURABH

```
main()
{
char s[10];
clrscr();
printf("enter your name");
scanf("%s", &s[0]);    → input: sau suk   output: sau
or
gets(s);               → input: sau suk   output: sau suk
puts(s);
getch();
}
```

क्योंकि scanf, space या enter को delimiter मानता है

same like scan — gets space को delimiter नहीं मानता बल्कि enter को मानता है

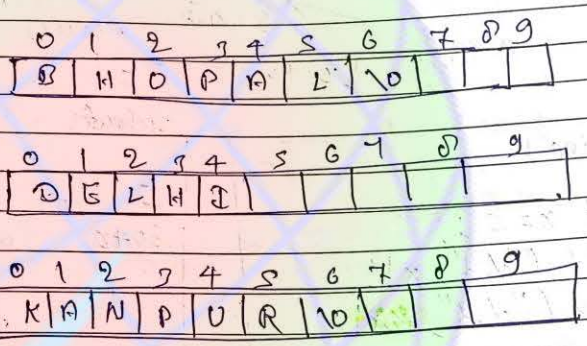
44: Lecture 13" String in C part 2

- strlen() – string length – single argument
- strrev() – string reverse – single "
- strlwr – string lower
- strupr – string upper
- strcpy – string copy eg. strcpy(s"Bhopal")
- strcmp – string comparision eg strcmp("AMAR", "AMIT")
  = ASCII value – ASCII value
- strcat() – string concatenation
  eg strcat(s, "student");

Charlie
Date
Page No.

* Handling multiple strings:-

• char s[3][10] = { "BHOPAL", "DELHI", "KANPUR" };



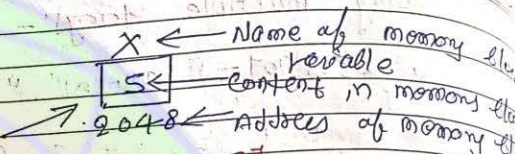
# main()
{
Char s[3][10];
int i;
clrscr();
Printf("enter three strings");
~~gets(&s[0][0]);~~
~~gets(&s~~
~~for~~(i=0 ; i<=2 ; i++)
gets(&s[i][0]); //gets(s[i])

for(i=0; i<=2; i++)
printf("%s\n", s[i]);
getch();
}

Pointers

45 ★ Lecture 14: Part 1: Pointer in C

1) `int x=5;`



memory block [0 to 65535]

```
main()
{
int x=5;
Printf("%d", x);
Printf("%d", &x);
}
```

output:
5
2048

2) **Address of operator (&) / Referencing operator**

- "&" is also known as address of operator
- It is Unary operator → अपना काम करने के लिए एक ही operand चाहिये।
- Operand must be the name of variable
- "&" operand gives address number of variable
- "&" is also known as referencing operator

3) **Indirection Operator (*) / Dereferencing operator**

- "*" is Indirection operator
- It is also known as dereferencing operator
- It is an Unary operator
- It takes address as an argument
- "*" returns the content/container whose address is it's argument.

Note- Simple बात याद रखो * , & का जरूर झगड़ा है।

Note: Unary operator Right to Left work करते है।

```
main()
{
  int x=5;
  Printf("%d", x);
  Printf("%d", &x);
  Printf("%d", *&x);
}
```

Output:

```
5
2048
5
```

(Simple rule: * और & जब भी आएँ दोनों को Cancel कर दो)

Note: %d ⇒ -32768 to 32767
%u ⇒ 0 to 65535 ← so, Pointers के case में %d के स्थान पे %u use की करते हैं क्योंकि address range ज्यादा है

1) Concept of asterick(*):⇒

x [5] 2048

int x=5;
&x=7; ← // error

"We cannot store anything in &x as &x is not a variable it is the way to represent address of block x"

• j=&x; // Compiler मे पता नहीं कर पायगा की मे address है या Constant

So,
int x=5; ← x एक ordinary variable है
int *j; ← j एक special variable है जो address contain करता है।
j=&x;

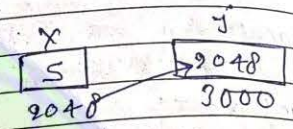
x [5] 2048 j [2048] 3000

और मे "j" ही pointer है।

We can store address in another variable
But j has to be declared before use

5) Pointer:-

```
int *j, x=5
j = &x;
```



- "j" is not an ordinary variable like any other integer variable.
- It is a variable which contents the address of another variable.

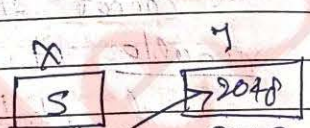
और से "j" की Pointer है, asterisk (*) लगा कर जो variable बना रहे है वही pointer variable है।

→ Pointer is a variable that contains address of another variable.

→ Pointer always consumes 2 bytes in memory

eg.

```
main()
{
int x=5, *j;
j = &x;
printf("%d %u \n", x, j);
printf("%d %u", *j, &x);
printf("%u", *&j);
}
```



Output

```
5 2048
5 2048
2048
```

6) Base add

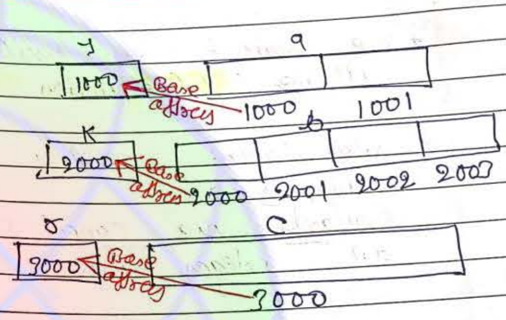
```
int a, *
float b,
char
j=&
#
```

6) Base address:-

int a, *j;

float b, *k;

char c, *r;



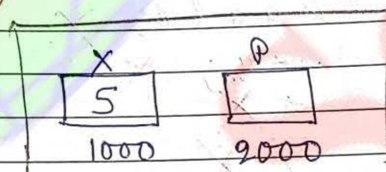
j = &a; k = &b; r = &c;

16# Lecture 14: Pointers in C part 2

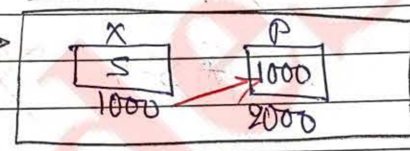
Objective

Extended concept of pointers:-

- void main()
  {
  int x = 5, *p; →



p = &x; →

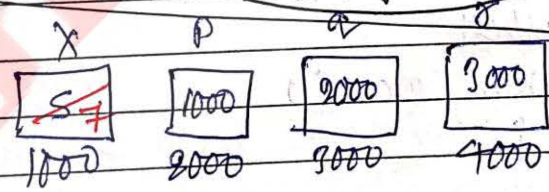


}

- void main()
  {
  int x = 5, *p, **q, ***r;
  p = &x;
  q = &p;
  r = &q;
  ***r = 7;
  }

(level of indirection fixes है मतलब **q सिर्फ *p का ही address रख सकता है। मतलब अपने से एक कम वाले का।

reason → r is a pointer to a pointer to a pointer to an int

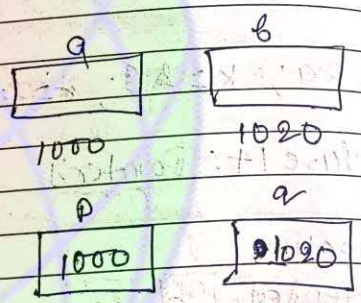


WWW.GRADESETTER.COM

Pointers Arithmetic

- We cannot add, multiply or divide two addresses (Subtraction is possible)
- We cannot multiply an integer to an address and similarly we cannot divide an address with an integer value.

```
void main()
{
int a, b;
int *p, *q;
p = &a;
q = &b;
```



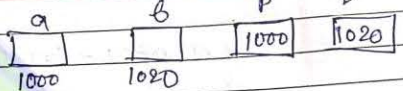

```
& a + & b;   X
p + q;       X
*            X
/            X
```

```
& a * 5;     X
p * 5;       X
/            X
```

- We can add or subtract integer to/from an address

```
void main()
{
```

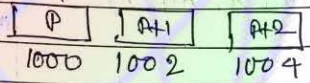
```
int a, b;
int *p, *q;
p = &a;
q = &b;
```



p+1 ⟶ 1002

↳ यदि pointer integer type का है तो अगर हम pointer में add करते हैं तो अगले block का address जाता है / अगले variable का address जाता है

Note:



~~pointer + n = pointer + size of (type of pointer) * n~~

Pointer + n = Pointer + size of (type of pointer) * n

1000 + 2 * 1

= 1002

P+1

P+4 ⟶ 1000 + 2 * 4 = 1008

int = 2

float = 4

P-1 ⟶ 998

Note: (Pointer arithmetic)

q - p ⟶ 1020 - 1000 = 10 ← No. of blocks

↳ Type should be same

Pointer 1 − Pointer 2 = $\frac{\text{Literal subtraction}}{\text{size of (type of pointer)}}$

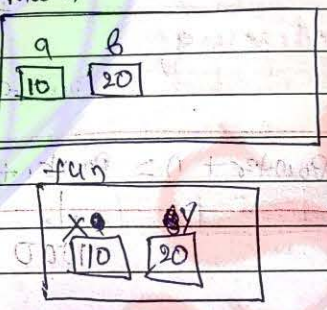
q − p ⟶ 1020 − 1000 = $\frac{20}{2}$ = 10 (int)

47# Lecture 15: Appli

Actual argument formal argument

pointer to function

```
void fun(int x, int y);
main()
{
int a, b;
---
fun(a, b); // actual arguments a, b
---
}
void fun(int x, int y) // formal arguments x, y
{
---
}
```



Q.) write a function to swap two integers

```
void swap(int *, int *)
main()
{
int a, b;
clrscr();
printf("enter two numbers");
```

```
scanf("%d %d", &a, &b);
swap(a, b);
printf("a = %d  b = %d", a, b);
getch();
}
void swap(int x, int y)
{
int t;
t = x;
x = y;
y = t;
}
```

Wrong हम कोशिश कर रहे थे कि x और y का value तो change हो जाएगा पर a, b का value change नहीं हो रहा है
So हमें pointer का use करना पड़ेगा ||

```
swap(&a, &b);
printf("a = %d  b = %d", a, b);
getch();
}

void swap(int *x, int *y)
{
int t;
t = *x;          // *x का मतलब a ही है
*x = *y;
*y = t;
}
```

output

```
Enter two numbers
10 20
a=20 b=10
```

Date
Page No.

* Call by reference / Address } This is same in C language but different in C++

- Call by reference is same as call by address
- when formal arguments are pointer variable, it is call by reference
- Reference means address

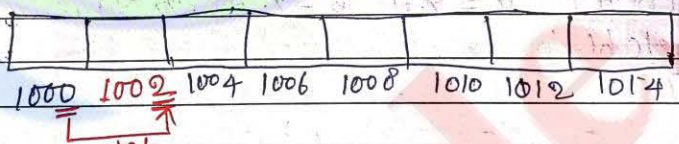
Q.) Why we use address of (&) in scanf()?

48# lecture is Application of pointer in c part 2 hindi

- Array always consume memory location in contiguous fashion

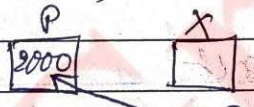
int a[8];

Application of pointer in Array



- Pointer when incremented, always point to immediately next block of it's own type

int x, *p; p=&x; p=p+1;



2000 + 2 = 2002

(int)

eg.)

main()
{
int i,
p = &
ter (i
scanf
ter (
Point
}

* Bubl

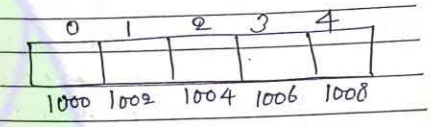
C language in C++ address variable,

2.

2 hindi

eg.)

```
main()
{
int i, a[5], *p;
p = &a[0];
for (i=0 ; i<=4; i++)
scanf("%d", p+i);
for (i=0 ; i <=4 ; i++)
Printf ("%d ", *(p+i));
}
```



* Bubble sort :-

```
void input (int *p)
{
int i;
for(i=0 ; i<=4; i++)
scanf("%d", p+i);
}

void display(int *p)
{
int i;
for(i=0; i<=4; i++)
printf ("%d", *(p+i));
}

void sort(int *p)
{
int round, t, i;
for( round=1; round<=4; round++)
{
```

```
for (i=0; i<=4-row; i++
    if (*(p+1) > *(p+i+1))
    {
      t = *(p+i);      // a[i]  *(p+i)
      *(p+i) = *(p+i+1);
      *(p+i+1) = t;
    }
```

```
main()
{
int a[5];
clrscr();
input(a);
display(a);
sort(a);
display(a);
getch();
}
```

output

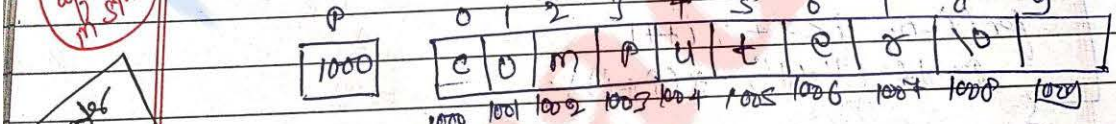
```
99
56
11
90
5

99 56 11 90 5  56 11 99 56 90
```

49# Lecture is Application of pointer in C Part 3:

(1) char pointers:-

Application of pointer in string

- string is stored in char array
- char s[10] = "Computer";

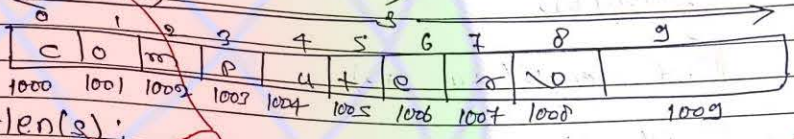


pointer + string

- char pointer can point to char block
- char *p;
  p = &s[0];
- *(p+i) or s[i]

(2) string Constant

- string Literal = string constant = string
- Printf ("Computer");
- char s[10] = "Computer";
- Printf(s);



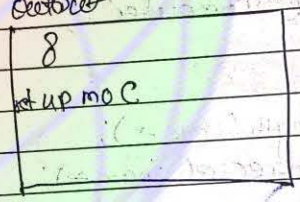
इसका मतलब इसका address पास किये।

- strlen(s);
  or
  strlen(&s[0]);
  or
  strlen("Computer");

```
int length(char *);
char* reverse(char *);
main()
{
clrscr();
Printf("%d", length("Computer");
Printf("\n%s", reverse("computer");
getch();
}

char* reverse(char *p)
{
int l, i;
char t;
for(l=0; *(p+l)!='\0'; l++);
for(i=0; i<l/2; i++)
  {
    t = *(p+i);
```

```
    *(p+i) = *(p+l-1-i);
    *(p+l-1-i) = ch;
  }
  return(p);
}
int length(char *p)
{
  int i;
  for(i=0; *(p+i)!='\0'; i++);
  return(i);
}
```

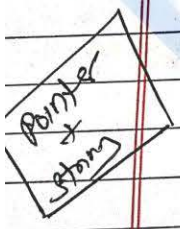
(l -> length of s)



So# Lecture 15 Application of pointer Part 4

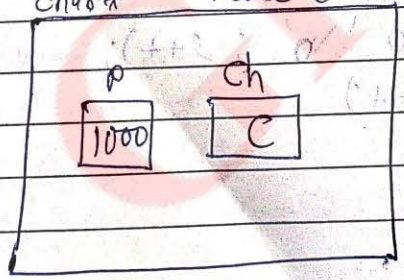
- Pointer and string

Program: write a function to reverse a string



0	1	2	3	4	5	6	7	8	9
C	o	m	p	u	t	e	r	\0	
1000	1001	1002	1003	1004	1005	1006	1007	1008	1009

```
char* reverse(char *p)
```



Charlie
Date
Page No.

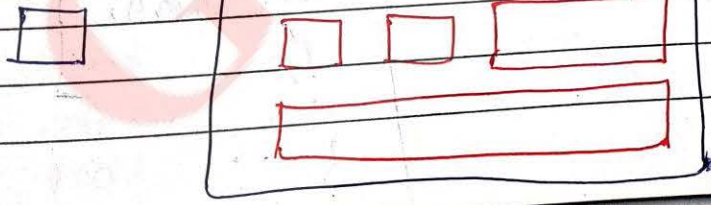
```
/* Function to reverse a string */

char * reverse(char *)
main()
{
clrscr();
printf("%s", reverse("computer"));
getch()
}

char * reverse (char *)
{
int i, l;
char ch;
for( l=0; *(p+l) != '\0' ; l++);
  for( i = 0 ; i < l/2 ; i++)
    {
    ch = *(p+i);
    *(p+i) = *(p+l-1-i);
    *(p+l-1-i) = ch;
  }
return(p);
}
```

Output

retupmoC

Sl# Lecture 16: Structure in C - Part-1

(1)
- Structure is a way to group variables
- Structure is a collection of dissimilar elements
- Defining structure means creating new data type.



2) Defining a structure

```
struct tag
{
// variable declarations here
};
```

```
# struct date
{
int d,m,y;
};
```

It not consumes memory

```
struct student
{
int rollno;
char name[20];
int age;
};
```

No memory is consumes

3) memory consumption

- No memory is consumes for definition of structure

4)

```
struct date
{
int d,m,y;
};
void main()
{
}
```

Global definition of structure

```
void main()
{
struct date
{
int d,m,y;
};
}
```

Local definition of structure

Charlie
Date
Page No.

5) struct date
{
int d,m,y;
}; ← global variable
struct date d1; ← global variable
void main()
{
struct date today,d1; ← local variable
today.d = 26; today.m = 7; today.y = 2015;
(or) struct date today = {26, 7, 2015};

today

d	m	y
26	7	2015

d1.d = today.d;
d1.m = today.m;
d1.y = today.y;
(or)
d1 = today;

d1

d	m	y

printf("enter today's date");
scanf("%d %d %d", &d1.d, &d1.m, &d1.y);
printf("Date: %d-%d-%d", d1.d, d1.m, d1.y);
getch();
}

output

Enter today's date 26/7/2015
Date: 26/7/2015

52# lecture 16: structure in C part-2

```
#include<stdio.h>
struct book
{
int bookid;
int title[20];
int price;
}
```

```
struct book input()
{
struct book b;
printf(" Enter bookid, title and Price");
scanf ("%d" &b.bookid);
fflush(stdin);
gets(b.title);
scanf("%f", &b.Price);
return(b);
}

void display(struct book b)
{
printf("\n%d %s %f", b.bookid, b.title, b.Price);
}

void main()
{
struct book b1;
b1 = input();
display(b1);
getch();
}
```

output

```
Enter bookid, title and
price
101 Drillingc 345.8
101 Drilling c 345.569998
```