Subject CSE1121

Monday

Date 28 01 19

Structural Programming Language

Theory - 3 1121.

75 marks →

→ Semester final (52.5)

→ C.I (15)

→ Atten (7.5)

LAB - 2 1122

25 marks

→ Practical (15)

→ Q, C.T (7.5)

→ Atten (2.5)

Computer

⇒ Programming Language.

⇒ Machine language

⇒ Algorithm: Steps of instruction of program

⇒ Flowchart.

Computer is a electronic device which can perform any logical or arithmetic operation via programming.

1st step of programming is Algorithm

↓using

~~2nd~~ Program Lang.

Scanned with CamScanner

Subject CSE 102/112 | Date 29 01 19

Main Components of computer.

* processor → takes all work load. VLSI - very large scale intregated chip.
* Motherboard - many components embeded on the board, IC Slots Socket Port.
* Memory
  - Primary (RAM) volatile
  - Secondary (HDD) non-volatile.
* Casing
* ROM → read only memory.
* I/O devices input/output/peripherial device
* PSU Unit → power supply unit → provides & controls power for all
* Heat sink (one kind of gel/aluminium used sit).

Generation of Comp.

1st: Vaccum tube.

2nd: Transistor

3rd: Iintegrated Circuit.

4th: LSI → large scale intregated

5th: VLSI eg. smartphone

Scanned with CamScanner

Subject

Date

```
#include<stdio.h>
int main()
{int x=10;
 float y=7.8;
 printf("%d\n", x);
 printf("%f\n", y);
 return 0;
}
```

Output:

10
7.8

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

Variable decleanation syntax:-

datatype variable name;

```
int x;
float y;
```

P-20: Programming exercise

CH-1 : 1.1, 1.2, 1.4, 1.5, 1.6

Scanned with CamScanner

Subject | Date

ax^2+bx+c . $x=5$ lower case.

```
int a, b, c, x;
scanf("%d", &a);
```

1

Algorithm:

Step 1: Start

Step 2: Decleare variables a, b, c, ~~and~~ x and result.

Step 3: Read values of a, b, c, x

Step 4: Find result = ax^2+bx+c .

Step 5: Display result

Step 6: ~~End~~ End.

```
int a, b, c, x, result
```

Scanned with CamScanner

Jency - 113

Subject CSE1121

Date 02 02 19

$ax^2 + bx + c$

Algorithm:

Step-1 : Start

Step-2 : Decleare variables a, x, b, c and result.

Step-3 : Read the values of the variables a, x, b and c.

Step-4 : Find the result, result = a*x*x + b*x + c

Step 5 : Display result

Step-6 : End.

Good

Program:-

```
#include <stdio.h>

int main()
{
    int a, b, c, x, result;
    scanf("%d\n %d\n %d\n %d\n", &a, &b, &c, &x);
    result = a*x*x + b*x + c;
    printf("%d\n", result);
    return 0;
}
```

Scanned with CamScanner

C.W.

Subject CSE1121

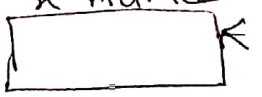
Date 02 02 '19

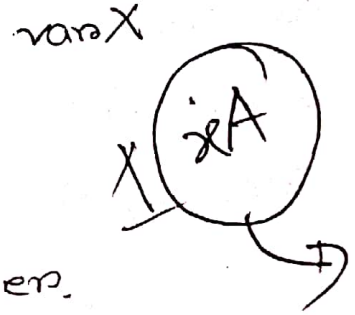
Variable

Declaration of variable

Rules of variable

Datatype
↳ memory size
&
↳ types of data.

int x
x named

memory address
2 byte



Datatype variablename ;
↳ cannot start with number.
2xX
x2✓
$, x_y
1st character of variable is alphabet.

H.W- Pg-25 : Table 2.3.
& 29 : Variable.

2*3/4+4/4+8-2+5/8
6/4+1+8-2+0
4+1+8-2
+1+8-2
+1+
-

Subject

Date

Write the program to display the following equation of st. line, $ax+by=c$. where,

$a=5$, $b=8$ and $c=18$.

Algorithm:-

1 Start ;

2 Declean variables a, b, c .

3 Read values of $a=5$, $b=8$, $c=18$.

4. Display st. line equation with values of a, b, c as $5x+8y=18$.

5. End

Program:-

```
#include <stdio.h>

void main()
{ int a,b,c;
   a=5;
   b=8;
   c=18;
printf("The equn of st. line is \n");
printf("%dx + %dy = %d", a,b,c);
getch();
}
```

Scanned with CamScanner

Subject CSE1121 | Date 04 02 19

CH-2, Constants, Variables and data types.

Character Set.

The C Character Set.

- C uses the uppercase letters A to Z.
- The lowercase letters a to z.
- The digit 0 to 9.

Rules of Identifier

1. 1st character must be an alphabet (or underscore).
2. Must consist of only letters, digit or underscore.
3. Only first 31 charetors are significant.
4. Cannot use a keyword.
5. Must not contain white space.

int a; — 2byte

float a; — 4 byte.

Scanned with CamScanner

a = 15
b = 71
c = 0
d = 21

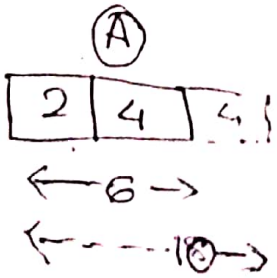
Derived datatype
1. Array
2. String
3. Structure
4. Union.

```
#include <stdio.h>
int main(){
int a,b,c,d;

a=15;
b=71;
c=0;
d=21;

printf("%dx + %dy - %dz = %d", b,a,c,d);
return 0;
}
```

8 4 2 1
1 0 0 1 = 9

(b)x + ay - cz = d
71x + 15y - 0.z = 21

derived
struct A
{
int a=5;
float b=2.5
float c=3.6;
}



int a=2
2 byte

datatype variable_name;
Example. int abc;

A variable is a named location in memory that is used to hold a value that can be modified by the program, All variables must be declared before they can be used.

Scanned with CamScanner

Subject | Date

int a,b,c;

variable name	valid?	Remark
First_tag	valid.	
char	no	keyward.
Price$	no	$ sign illegal.
group one	no	no blank space
1 2 3 4 5 6 7 8 average_number	valid.	1st 8 charecters are significant
int_type	valid.	

a=5, b=6.

① swapping number. ② ASCII value.

Algorithm:-

Step-1: Start.

Step-2: Decleare variable a, b.

Step-3: Read the values of ~~a=5~~, variables a=5, b=6.

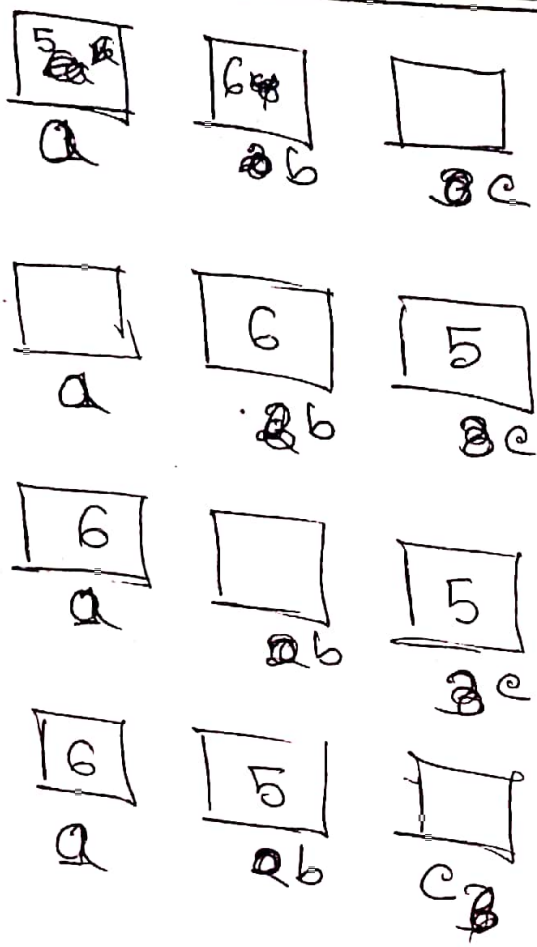
Step-4: Display a=5, b=6.

Step-5:

Scanned with CamScanner

Subject

Date



int a=5;
int b=6;
int c;
temp
c=a;
a=b;
b=c;

assignment operator. left to right.

```
#include <stdio.h>
int main(){
    int a=5;
    int b=6;
    int temp;
    temp = a;
    a = b;
    b = temp;
printf("Swapped value a=%d b=%d", a, b);
}
```

Scanned with CamScanner

Subject

Jency

Date

```
#include<stdio.h>
int main(){
    int a,b,temp;
    printf("Input value of a = ");
    scanf("%d", a);
    printf("Input value of b = ");
    scanf("%d", b);
    temp=a;
    a=b;
    b=temp;
    printf("swapped value a = %d, b = %d", a,b);
    }
```

Enter the value of a=

u u u u b

Before swapping

a=

b=

After swapping the value of

a=

b=

Scanned with CamScanner

Constant

'A'

Single charecter Constant → single char in Single quote ' '

String Constant → Sequence of charecter in double quote " "

"AaBb"

Q: Differenciate betw 'A' "Aa".

'A'	"Aa"
1) Called Single charecter constant	1) called string constant
2) Single char	2) Sequence of many character
3) Bounded in single quote	3) Bounded in double quote.
4) Can give / Call	

%c → character

%s → string.

%d → integer

Output:

5

65.

c

Hello

```
int a=5;
int x='A';
char y='c';
String a1="Hello";
printf("%d\n", a);
printf("%d\n", x);
printf("%c\n", y);
printf("%s\n", a1);
```

Scanned with CamScanner

Subject CSE 1121 | Date

C OPERATORS, OPERANDS, Expression & STATEMENTS.

80-100 -> 4
75-79 -> 3.75
70-74 -> 3.5

* LAB Test-

1. Swapping.
2. Equation display $ax+by+c=0$.
3. Result of sum, result $= ax^2+bx+c$
4. Float & integer number display.
5. Area of triangle
6. Sum of 2 number

Based on arity, operators are classified as nullary (no operands), bi tri

Task.

days = 265
month = days / 30;
days = days % 30; 25

month = 0 ,
days = 0.83

Find out the value of days and month.

months = 0.833
days =

Subject

Date

* Arithmetic operators.

c = 2
a = 100
b = 5

5/2
= 2) 5 (2.
4
1

| + / * |
| - % |

arithmetic

eq = a + b % c
=
= 101 .

Relational Operator:- The relation bet^w two operands.

x > y. < > = <= ==
x == 5

Logical operator: 3

&& - logical AND

|| - logical OR.

!= - ,, not

Salary = 1000 . 1 = 0.

if ((study > 1000) && (salary < 2000))

printf (a

Scanned with CamScanner

Subject | Date

if (f >= 18 && m >= 21)

if((f >= 18) && (m >= 21))
    printf ("Yes");

int age = 100;

if ((age >= 80) && (age <= 100))
printf ("you are getting older");

Assignment Operator :- use → i) To assign a value to a variable.

age = 80
(= assignment Operator)

Shortend form:

x = x + y
x += y

Increment, Decrement
Unary → ++ , --

x++ → postfix आधा Print then increase +1.
++x → prefix " increase +1 then display

x-- → postfix
--x → prefix

x = 5

printf("%d\n", x++);
5
printf("%d\n", x++);
6

Scanned with CamScanner

Subject

Date 16 02 19

Increment/Decrement operator. → ++ -- it is unary operator,

Conditional operator.

Comma operation

Bitwise operator

Size of operator.

Arithmetic operator.

and its precedence

'++' operator add 1 to the operands.

'--' operator substract 1 to the operands.

post / pre > increment/ decrement.

++x > pre increment

x++ > post "

post → 1st evaluates the original value.
2nd it increments the value.

```
int a=15;
int b=10;
int c = ++a - b;
printf("%d %d %d", a, b, c);
```

11
15
26

16 10 6

```
int d = ++b + a++;
```

11. 15

```
int x=5;
int y=6;
int z;
printf("%d", x++);
printf("%d", ++x);
printf("%d", x);
printf("%d", ++x);
```

5 6
6 7
7
5
6
7
8

Scanned with CamScanner

Subject | Date

5
6
6 6 6
6 7

x++ = 5+1 = 6

x++ = 6 +1 = 7

x = 7 +1 = 8

++x = 6+1 = 7

6
7
8

(x>y) ? x : y
y

CONDITIONAL OPERATOR

if (condition) — true (~~Trinary~~ Ternary operator ?:)
// statement execute.
else
// statement.

Syntax:
exp1 ? exp2 : exp3

~~if (x>5)~~ x=100
Y=10

if (x>y)
printf("x larger");
else
printf("y larger");

16 - 5 = 10

int a=5, b=6, c;
int x;
c = ++a + 10 - b--
x = (a>b) ? 1 : 0; (=0)
x = (a<b) ; a : b;
Printf("%d", c);
Print F("%d", x);

10
0/a

Scanned with CamScanner

Subject

Date

Sizeof() operator (compile time)

int sum;
↓
2 byte

Size of (sum);

Arithmetic expresion:-

higher priority: *, /, %

lower " : +, -

associativity : L to R.

```
for(m=0; m<3; ++m)
    (m%2) ? m : m+2 .
```

m = 0

0%2 → False 0+2 = 2.

m = 1

1%2 → 1 (fraction) 1

2+2

H.W: Assignment : Monday.

(3.10 Pg (77-78)
-3.17)

Debugging exercise practice (3.1 - 3.3).

Scanned with CamScanner

Subject

Date

\n

/*

puts

paro virus

//

20.02.19

loop ⇒ While

initialization

while (condition)

{

increment/decrement;

}

for

for(initialization; condition; inretation)

*/

* year=0, amount= 5000.00

value = amount + inrate * amount

= 5000 + 0.11 * 5000

= 5000 + 550

= 5550

* year = 0 + 1 = 1 amount = 5550.00

value = amount + inrate * amount

= 5550 + 0.11 * 5550

= 5550 + 610.5

= 6160.50

* year = 1 + 1 = 2 amount = 6160.50

value = 6160.5 + 0.11 X 6160.5

= 6160.5 + 677.6

= 6838.15

* year = 2 + 1 = 3 amount = 6838.1

Scanned with CamScanner

Subject | Date

~~* year = 3+1 = 4 amount~~

value = 6838.1 + 0.11 X 6838.1

= 6838.1 + 752.191 = 7590.35

* year = 3+1 = 4 amount = 7590.35

value = 7590.35 + 7590.35 X 0.11 = 7590.35 + 834.93

Monday - C.T. - 25.02.19.

CH - 1, 2, 3

Total Mark = 25.

* Question-Answer.
* Error & Finding.
* ~~be~~ debugging solution.

Type Conversion (data type) int, float : double

- Implicit type conversion
- Explicit "

Scanned with CamScanner

Subject

Date

Implicit type conversion : automatically change the output.

#int x

x = 3.5;

output: 3

Explicit type conversion : by force.

float x = 10.7;

= int 10.7;

Operator Precedence : Order or Priority to operator.

Precedence : Order in which different operators are applied

expression

Associativity : Order in which multiple occurrences of same level operator are applied.

a = 5

R to L

assignment

a == 5

L to R

equality.

Subject CH-4

Date

Managing I/O Operation

Reading → Input

Processing

Writing → Output.

{scanf()
printf()

#include<stdio.h>

{getchar() ⇒ reading a char
putchar() ⇒ writing a char.

Syntax :

variable_name = getchar();

putchar(variable_name);

Program:

```
#include<stdio.h>
int main()
{
  char x;
  printf("if your name tanvir input
    'Y' or 'y', otherwise
    press 'N' or 'n'");
  x = getchar();
  if (x=='Y'||x=='y')
    printf("I am tanvir");
  else
    printf("I am not
      tanvir");
```

ctype.h → character func a A 1 a

buildin

isalpha() → is it an alphabet

Isalnum() → is it an alpha numeric

is digit() → is it a digit

Islower()

is print()

is punct()

is space()

isupper()

islowered

to upper()

to lower()

```
char x
 x = getchar()
if (isalpha(x) > 0)
```

condition

value — T → 1

Scanned with CamScanner

Subject

Date

Decision Making statement.

(i) if

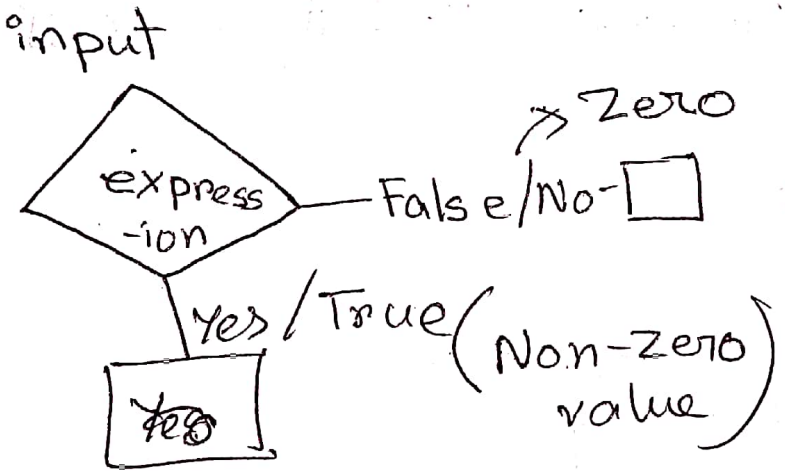
(ii) Switch

(iii) Conditional

(iv) go to

Syntax:

if (condition)




```
int i=-10
  if(i) → non-zero } True output statements.
        → true
```

```
i=0
  if(i) → False } no output of inside statement.
    0 means ↑
```

Scanned with CamScanner

tion

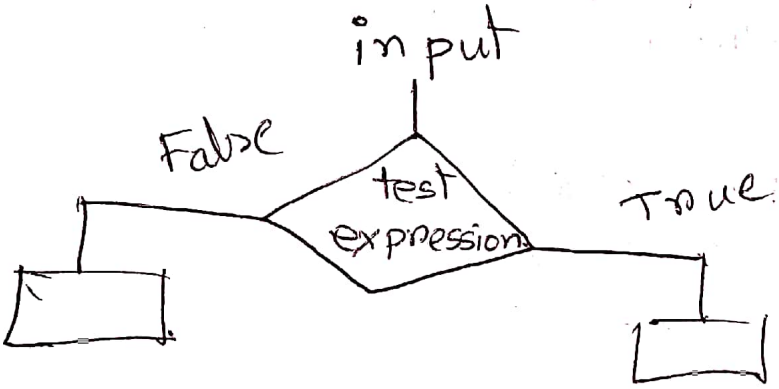
Subject

Date

The if.....else statement.

```
if (expression)
   //statement.. statisfied.
else
   //statement.
```

```
String c = getchar()
   if (c == "CSE")
        printf ("you are a student of CSE")
```



a → largest a > b && a > c

b → b > a && b > c

c → ~ largest.

Scanned with CamScanner

Subject

Date

```
if()
  // statement
else { }
```

```
if (condition 1)
  // statement 1
else if (condition 2)
  // statement 2
else if (condition N)
  // statement N
else
  // default statement.
```

Algorithm for getting highest number.

Step-1: Start.

Step-2: Declear variables. a, b, c.

Step-3: Get ~~input~~ variables ~~input~~ input.

Step-4: If a > b && a > c display a as the greatest
else if b > c && b > a display b as the
else print c as largest.

Step-5: end

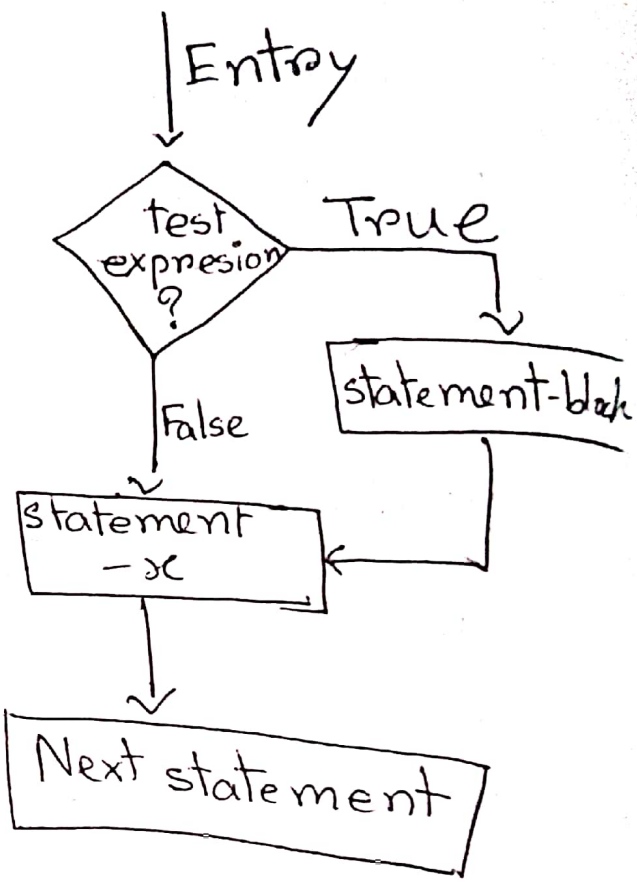
Scanned with CamScanner

Subject @SE1121

Date 02 03 19

Simple if-Statement.

1a) if (test expression)
{
statement-block;
}
statement-x;



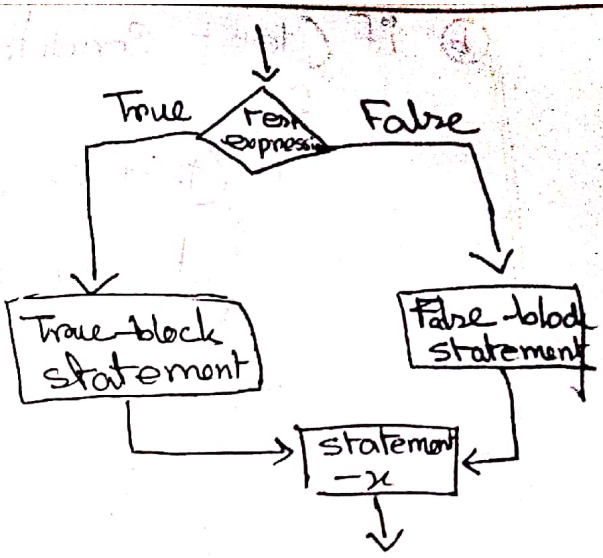
1b) if (test expression)
{
statement-block;
}
statement-x;

2.a) The if---else statement.

If (text expression)
{
True-block statement(s)
}
else
{
Fblse-block statement(s)
}
statement-x.

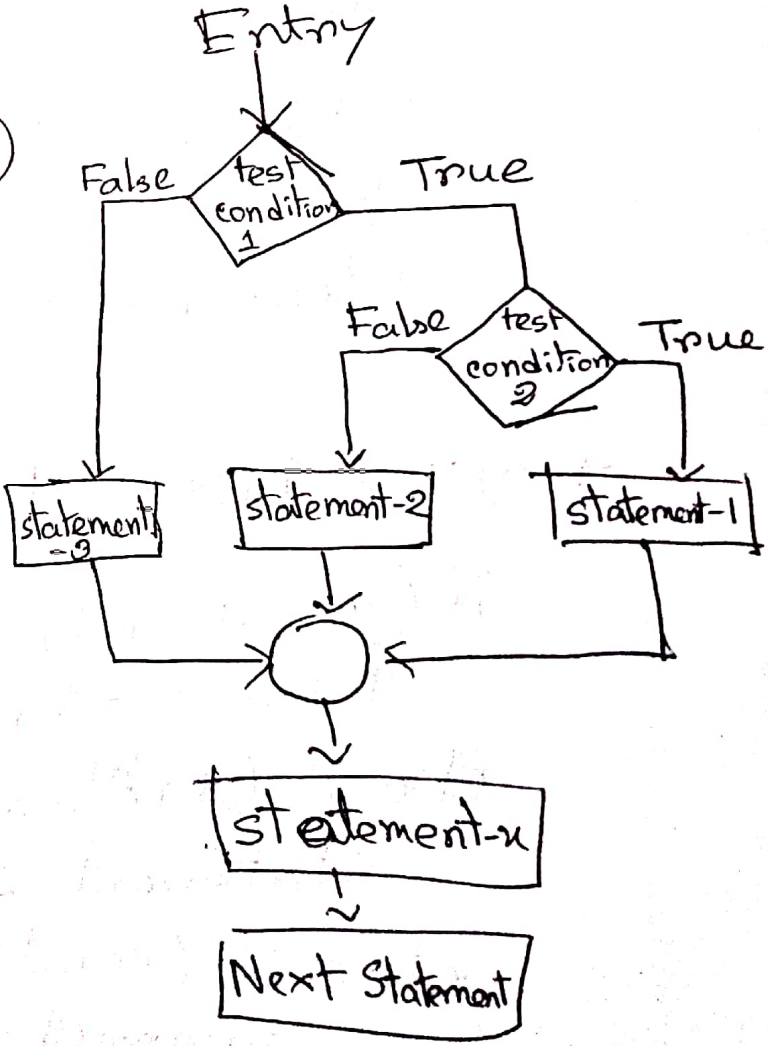
Scanned with CamScanner

2.b) if (test-expression)
{
True-block statement;
}
else
{
False-block statement;
}
statement-x



3. Nesting of If... Else statement.

a)
```
if (test-condition-1)
  { if (test condition-2)
      {
       statement-1;
      }
    else
      {statement-2;
      }
  }
else
  { statement-3;
  }
statement-x;
```



Scanned with CamScanner

(b)
```
if (test Condition-1)
 {
  if (test Condition-2)
    {statement-1;
    }
  else
    {statement-2;
    }
 }
else
 {
   statement-3;
 }
statement-n;
```

4. a) The Else-if ladder

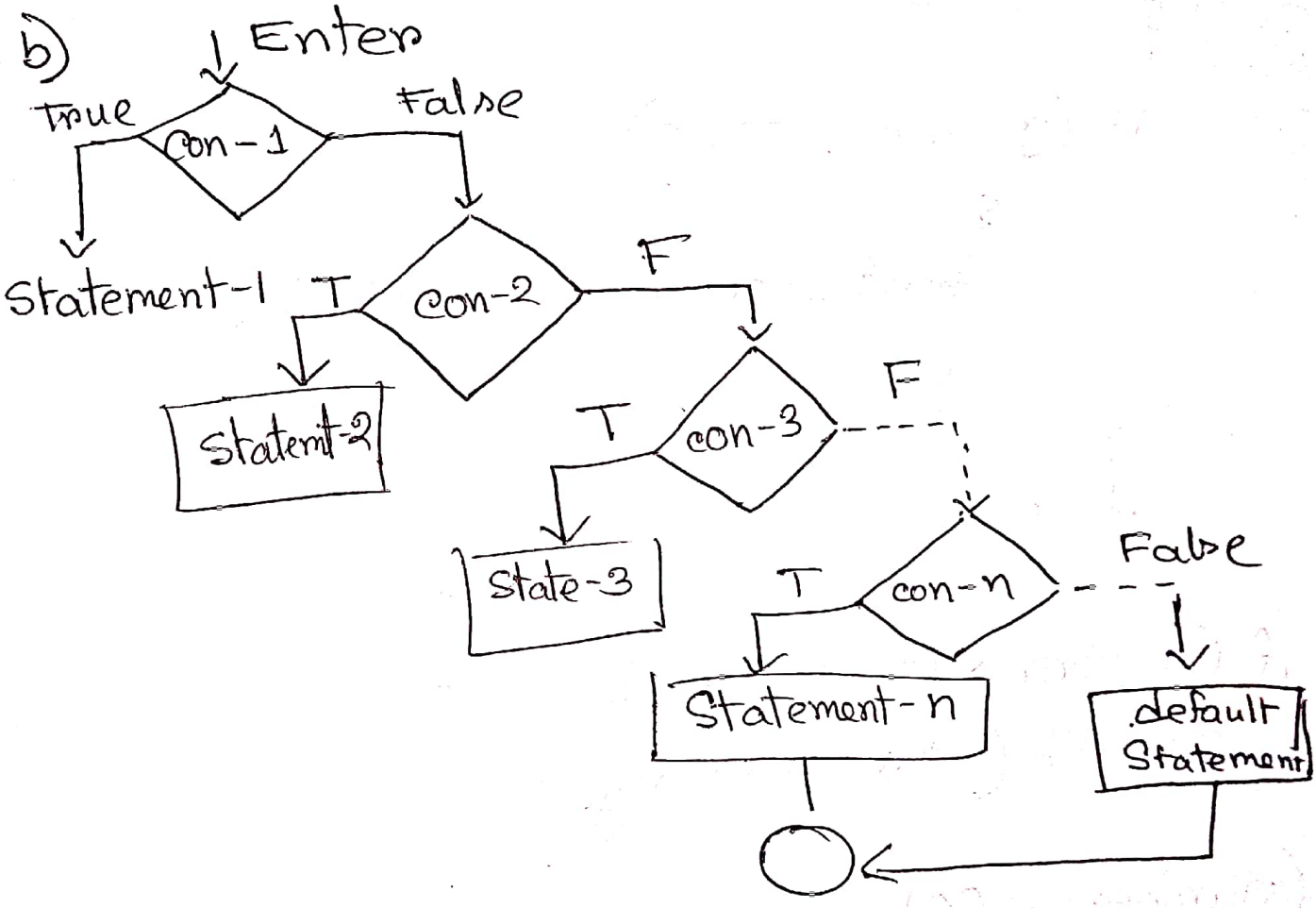
```
if (condition 1)
   statement-1;
else if (condition 2)
     statement-2;
else if (condition 3)
     statement-3;
else if (condition n)
     statement-n;
```

Scanned with CamScanner

else
default-statement;

statement-x;

b)



Scanned with CamScanner

a) if (grade <= 59 && grade >= 50)
    second = second + 1;

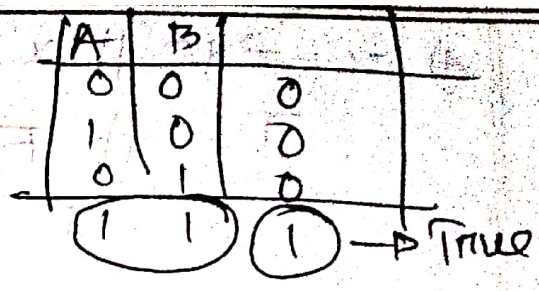
```
if (grade <=59)
    {
    if (grade >=50)
        {
        printf second = second + 1;
        }
    }
```

b) if (number > 100 || number < 0)
    printf ("Out of range");

```
if (number > 100)
    printf ("Out of range");
else if (number < 0)
    printf ("Out of range");
else
    sum = sum + number;
```

Scanned with CamScanner

d) if (M1 > 60)
{
if (M2 > 60)
printf ("Admitted \n");
else if (T > 200)
printf ("Admitted \n");
}
else
printf ("Not admitted \n");



5.5. x = 10.

a) x == 10 && x > 10 && !x
T && T && F
= F

b) x == 10 || x > 10 && !x
T || F && F
T || F = T

c) x == 10 && x > 10 || !x
T && F || F
T && F → F

Subject

Date

d) $x == 10 \;||\; x > 10 \;||\; !x$

$T \;||\; F \;||\; F$

$T \;||\; F = T$

5.6 $x=1$ & $y=2$.

a) switch (2) ⟶

b) case 10:

c) switch (3)

d) Switch (1)

```
{ case 2:
    y = x + y;
  break;
```

```
Swich ( )
  { case value 1:
              break;
    case value 2:
              break;
  } default:
              break;
```

Scanned with CamScanner

a) ! (x<=10)

Here, if "x less than or equal to 10" this condition is true, Then the ! sign indecates it will give a false output.

```
if ( a> 30)
{ if (b> 10)
  { printf ("
```

The statement will execute if x is not less than or equal to 10,

b) !((x==10) || !((y==5) || (z<0))

The statement will be true if x is not equal to 10 or y is not equal to 5 or z is greater than 0,

Scanned with CamScanner

Subject

Date

CH-6: Decision Making and looping. while, for, do...while

loop: performs a set of operation ~~which does some work~~ that ~~repeatly~~ repeadly execute the statement untill the control variable fails to satisfy the certain condition.

```
while (test-condition)
  {
    body of loop
  }
```

4 steps loops follow:-

(i) Setting and initialize control variable.
↳ decleared variable.
Summation (100 - 150) x=100

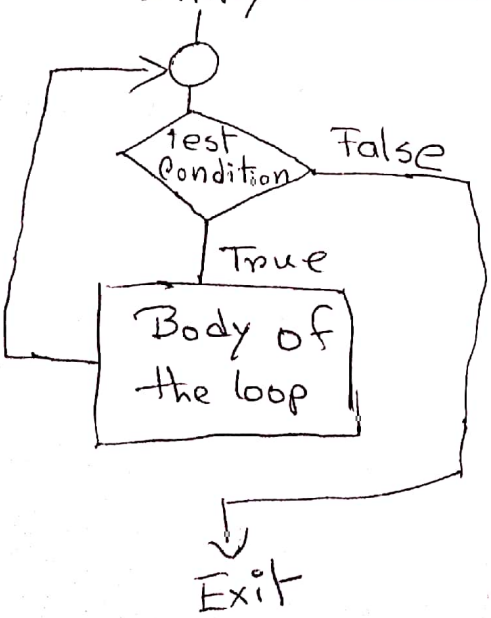
(ii) Execution the statement of the loop.

(iii) Test for a specific value of a control variable.

(iv) increment/decrement the control variable.

loop Control Structure

Entry Entry Controlled loop



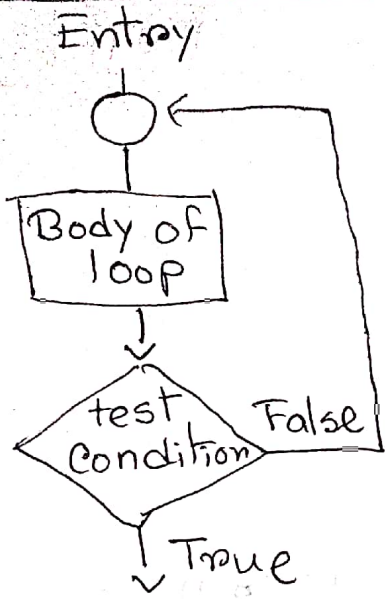
for, while → entry controlled.

Scanned with CamScanner

Subject | Date

Exit Controlled loop

1 2,3,4,5 6,7 8,9 10
billion



do...while → exit controlled.

For loop

depends on control variable

use conditional operator.

for(initialization; test condition; increment/decrement)
{
//loop body
}

~~initi~~

```
for (i=1; i<=10; i++)
{
printf ("%d", i)
}
```

output.

i=1	1<=10	→ 1
i++ i=2	2<=10	→ 2
i=3	3<=10	→ 3
i=4	4<=10	→ 4
i=5	5<=10	→ 5
i=6	6<=10	→ 6
i=7	7<=10	→ 7
i=8	8<=10	→ 8
i=9	9<=10	→ 9
i=10	10<=10	→ 10
i=11	11<=10 ✗	end.

Scanned with CamScanner

Subject

Date

```
for (x=9 ; x<9 ; x=x-1)
{
  printf("%d", x);
}
```

Output: NOThing

```
for (x=10; x<25 ; x=x+1)
  printf("%d", x)
```

10
11
.

```
sum=0.
for (x=10; x<25; x=x+1)
{
  sum=sum+x;
}
printf("%d", sum)
```

x=10, 10<25, Sum = 0+10 = 10

x=10+1 = x=11, 11<25, Sum = 10+11 = 21

x=12, 12<25, Sum = 21+12 = 33

x=13, 13<25, Sum = 13+33 = 46

x=14, 14<25, Sum = 14+46 = 60

x=15, 15<25, Sum = 60+15 = 75

x=16, 16<25, Sum = 75+16 = 91

x=17, 17<25, Sum = 91+17 = 108

x=18, 18<25, Sum = 108+18 = 126

x=19, 19<25 Sum = 126+19 = 145

x=20, 20<25 Sum = 145+20 = 165

x=21, 21<25, sum = 165+21 = 186

x=22, 22<25 Sum = 186+22 = 208

x=23, 23<25, sum = 208+23 = 231

x=24, 24<25, Sum = 231+24 = 255

Sum = 255

x=25 25<25 → F.

Scanned with CamScanner

Subject

Date

```
for (i=1; i<=2; i++)
  {
    for (j=1; j<=2; j++)
      {
      printf ("%d %d", i, j)
      }
  }
```

i=1

j=1, 1<=2 → T

Op → 1 1 j++

j=2, 2<=2 → T

Op → 1 2.

j≠3 ∵ 3<=2 → False

i=2 2<=2 → T

j=1, 1<=2 → T

Op → 2 1 j++

j=2, 2<=2 → T

Op → 2 2 j++

j=3, 3<=2 → F.

∴ Output:

11122122

Scanned with CamScanner

5.13

```
main()
{
    int m,n,p;
    for(m=0; m<3; m++)
    {
        for(n=0; n<3; n++)
        {
            for(p=0; p<3; p++)
            {
                if(m+n+p == 2)
                {
                    goto print;
                }
            }
        }
    }
    print:
    printf("%d, %d, %d", m, n, p);
}
```

Program Process:

① m=0
n=0
p=0 → T

② m=0
n=0
p=1 → T

③ m=0
n=0
p=2 → T

0+0+2 == 2

~~m=0~~
~~n=0~~
~~p=3 → F~~

Scanned with CamScanner

Subject

Date

(4) m=0
n=1
p=0

(5) m=0
n=1
p=1

(6) m=0
n=1
p=2

(7) m=0
n=2
p=0

(8) m=0
n=2
p=1

(9) m=0
n=2
p=2

(10) m=1
n=0
p=0

(11) m=1
n=0
p=1

(12) m=1
n=0
p=1

double

(13) m=1
n=0
p=1

(14) m=1
n=1
p=0

(15) m=1
n=1
p=1

(16) m=1
n=1
p=2

(17) m=1
n=2
p=0

(18) m=1
n=2
p=1

(19) m=1
n=2
p=2

(20) m=2
n=0
p=0

(21) m=2
n=0
p=1

(22) m=2
n=0
p=2

(23) m=2
n=0 1
p=2

(24) m=2
n=1
p=1

(25) m=2
n=1
p=2

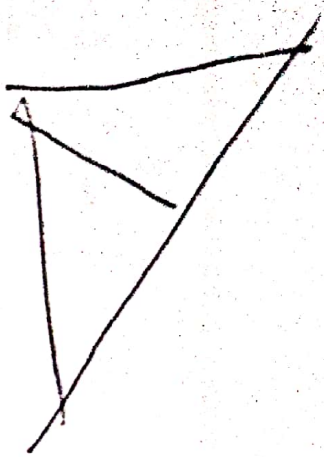
(26) m=2
n=2
p=0

(27) m=2
n=2
p=1

(28) m=2
n=2
p=2

Scanned with CamScanner

```
int a
float sum=0;
for(a=1; a<=99; a++)
  {
    sum=sum+(1/(float)a);
  }
printf("%f", sum);
}
```

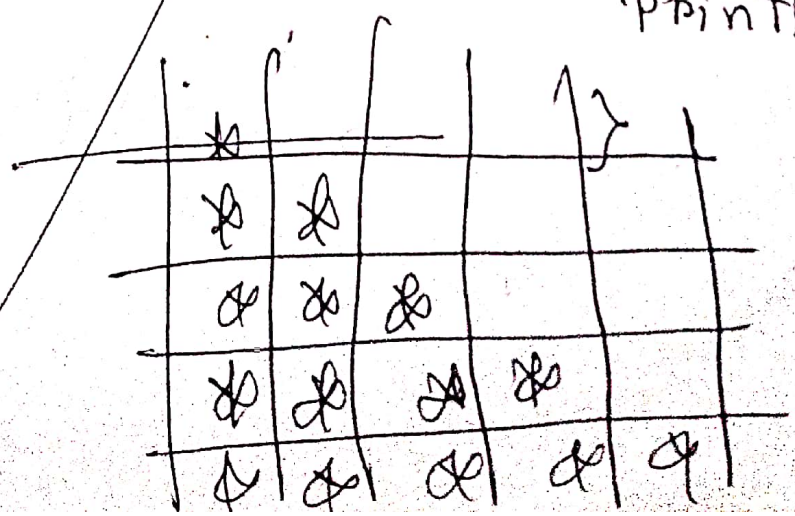


1

```
int a,b,c=1
for(a=1;a<5;a++)
{
 for(b=1; b<c; b++)
    printf("*");
    printf("\n");
    c++
  }
 return 0;
}
```

```
(1) int i,j
  for(i=1; i<=5; i++)
   {
→ for(j=1; j<=i; j++)
    {
    printf("*")
    }
  printf("\n");
```

i = row
j = colum



Subject

Date

i=1 1<=2

{

j=1 j<=i

1<=1 → True.

j=2 2<=1 → False.

---

i=2

j=1 1<=2 → T

j=2 2<=2 → T

i=3

j=1 1<=3 → T

j=2 2<=3 → T

j=3 3<=3 → T

5 4 3 2 1

1

1 2

1 2 3

1 2 3 4

① → printf("%d", j).

Scanned with CamScanner

Subject

Date

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

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

i = row
j = column

5
4
3
2
1
time

Print,

1. 2, 3 4 5
1 2 3 4

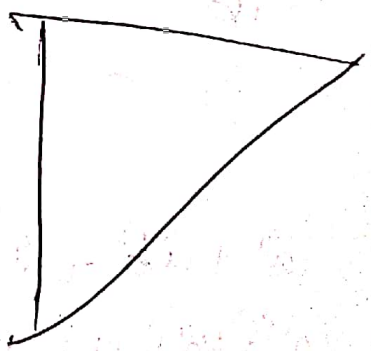
i = 5,
j = 1
j = 2
j = 3
j = 4
j = 5

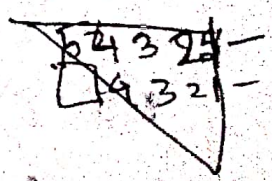
i = 4 → j = 1, 2, 3, 4

i = 3 → j = 1, 2, 3

i = 2 → j = 1, 2

i = 1 → j = 1



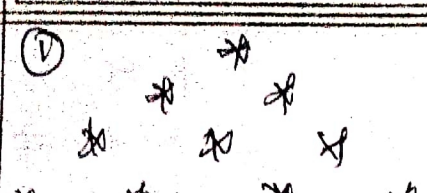


Scanned with CamScanner

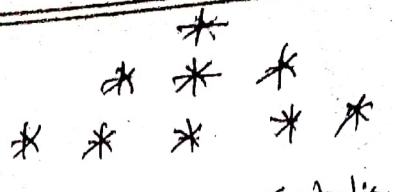
Subject

Date

(1)



```
int main(){
int r, sp, p, no, n;
printf("Enter no of rows: ");
scanf("%d", &no);
n=no;
for (r=1; r<=no; r++)
  {
   for (sp=1; sp<=n; sp++)
    {
     printf(" ");
    }
    n--;
    for (p=1; p<=r; p++)
    {printf ("* ");
    }
    printf ("\n");
  }
return 0;
```



```
#include <stdio.h>
int main(){
  int r, sp, p=0; ui;
  printf("Row no: ");
  scanf("%d", &ui);
  for(r=1; r<=ui; r++)
   {
    for (sp=1; sp<=ui-r; sp++)
    { printf(" ");
    }
    while (p != (2*r-1))
     {printf ("* ");
      p++;
     }
     p=0;
     printf ("\n");
   }
   return 0;
}
```

Scanned with CamScanner

Subject

Date

(2)
```
1
2 3
4 5 6
7 8 9 10
```

```
#include<stdio.h>
int main(){
  int r, p, ui, value=1;
 printf("Enter Number of row:");
  scanf("%d",&ui);
 for(r=1; r<=ui; r++)
 {
    for(p=1; p<=r; p++)
      {
        printf("%d", value);
        value++;
      }
      printf("\n");
  }
  return 0;
}
```

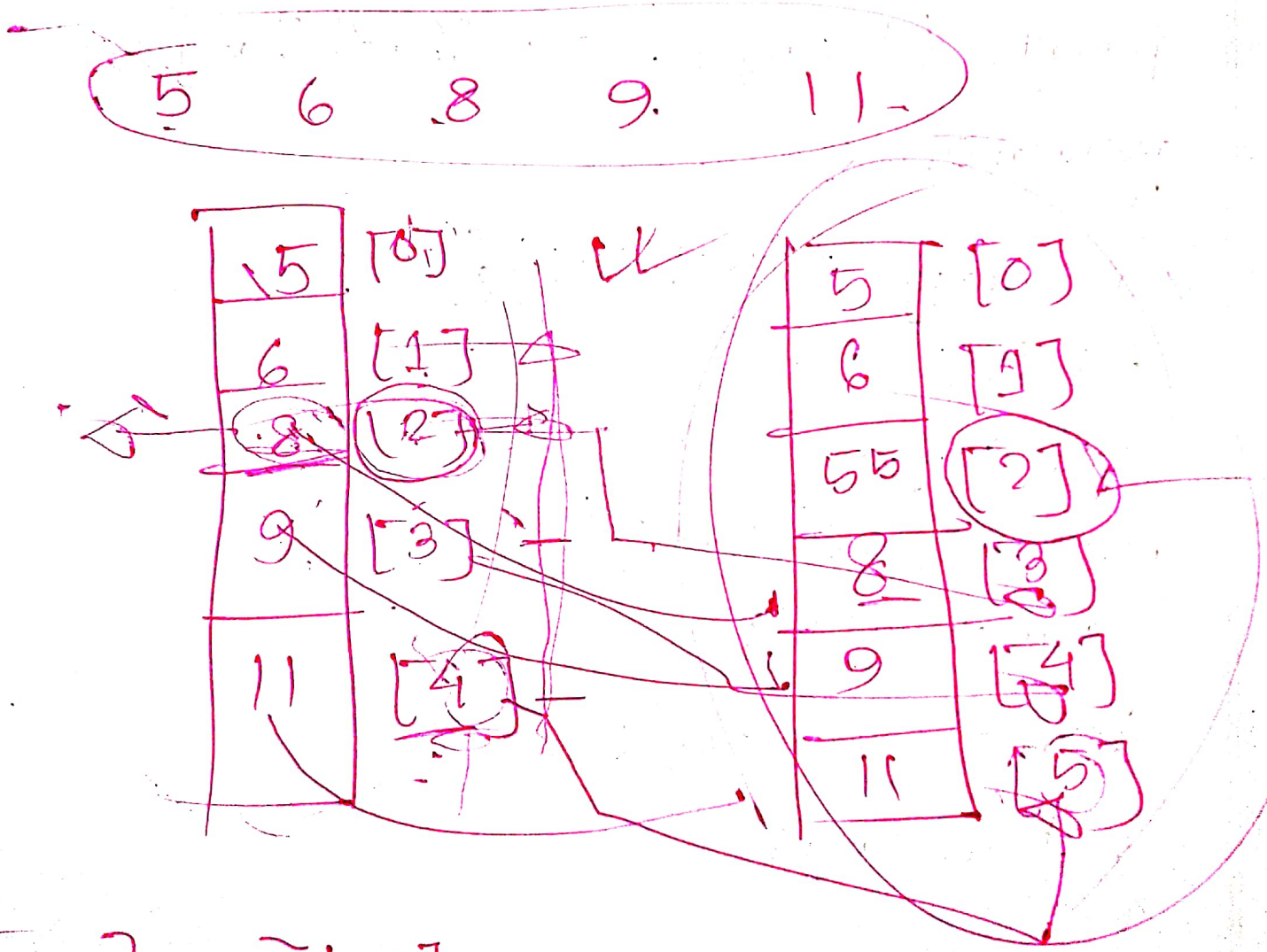
(3)
```
* * * * *
  * * *
    *
```

```
#include<stdio.h>
int main(){
  int r, ui, sp, p=0;
 printf("Enter N:");
 scanf("%d",&ui);
 for(r=ui; r>=1; r--)
  {
    for(sp=0; sp<=ui-r; sp++)
     {printf(" ");
     }
    p=0;
   while(p!=(2*r-1))
    {printf("* ");
     p++;
    }
    printf("\n");
  }
 return 0;
}
```

Scanned with CamScanner

Recursion:

Recursion defines a function in terms of itself. It is a programming technique in which a func calls itself. A recursive func calls itself repeatedly.



[ ]+1 = a[ ]

4-1=3

Scanned with CamScanner

Subject

Date

CH-2

1) Pg-22 - character set (defination)
2) C token & types of token
3) Keywords & name learn (32 টা)
4) What is keyword, identifier ? Rules of identifier
5) Types of token (defination + types + example)
6) Constant কত প্রকার & example.
7) ~~22~~ pg-28 table.
8) Difference between ~~sys~~ single charector constant, string character constant. (def^n + example)
9) Identifier & variable [Pg-29. Defination & rules]
10) Datatype (Defi. + examples. all datatype + memory size
11) int, Float, char, void (definition + example)
long int, int, short int.
12) Floating - long, double, float.
13) ~~Type~~ User define declaration Pg-35, কাকে বলে ? example ? কি কাজ

typedef enum

Scanned with CamScanner

Subject

Date

Page 42 ; What is symbolic constant ?
Rules & Reading 43 Page 9
Overflow & underflow 44 page
Page 48 : 2.4, 2.6, 2.7, 2.9, 2.12, 2.13, 2.19

H.W Page 50 : 2.15, 2.17

Chapter 3

Page 51 : What is operators, Definition, types of operators, Example
Arithmetic, Relational, Logical operator, Assignment operator (with example)
↓
Increament, decreament (example, difference
59 Page : Conditional operator (definition, chara, exmple)
Bitwise operator, definition
example

Page 60 : Size of operator

Page 60 : size of operator.

Page 66 : Type conversion (types & definition & description
implecite & explecite

Scanned with CamScanner

Subject | Date

Pg-69 Precedence & Associativity rules.

Pg-71 : Rules of Precedence & associativity

73 SALESMEN salary, quadratic equ^n

7 · 3·03 – 3·17

CH-4

Pg-883 · difference betw scanf(), getchar(), gets()
printf(), putchar(), puts() [syntex, defi, example.

4·1 → isalpha( ) → all.

Q: What is control string ? (defi. example)

Pg-86,

Reading for debugging soln.

Reading Inputting Real number (error finding)

Inputting Character string.

In exam: Output আকারে input program লিখতে হবে

Pg-90 4·6

Scanned with CamScanner

Subject

Date

Reading mixed data type Pg- 91,

Programming knowledge

Page - 93 %d, %f etc. table. → Programming skill

Formatted input, output theory.

Pg-86, 93 defi, example

VVI: Formatted input.

Pg - 95: Output integer number [H.W.]

Debugging exercise → board final.

Pg - 96: Program to Output → 4·9

Pg - 97, 98, 99 [output]

Prog - [4·11] → Output

key concept Pg-101, 102.

always remember

Scanned with CamScanner

S.T

16, 17, 18, 19

Subject

Date

Pg - 107 - 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12.

4.15

CH-4 → pre all program do, C

Debugging - 4.1, 4.6, 4.9,

CH-5

Pg - 111: What is decision making statement?

Example.

112. Simple if → flow dragram + syntex + example.

if else,

115 else if

118, 121, 5.5, 126 - Program (learning part)

goto statement, - 133

136 →

138 → Table & calulation.

146 → 5.1, 5.2, 5.3, 5.4, 5.5, 5.7, 5.9, 5.10, 13, 14

CH-6 : what is looping (150)

definition, type.

151 → Syntex, example, differences

while, do...while

comparism → for, while, do-while.

Scanned with CamScanner

Subject CH-7. Date

Array

fundamental data type: int, char, float, string

```
int x=5;
float y=10'5;
char xy='A';
string xz="ABC";
```

fundamental character: All datatype can store single data in single variable.

Q: Difference int x[10], int x;

int x=5;

the integer datatype variable x can store only one value at a time,

array: same datatype, more than one value can store sequentially in a memory

Scanned with CamScanner

Subject | Date

Array declearation syntex:

Datatype . arrayname [size]
↳ number of elements of same datatype

100, 1, 2, 3, 5, 6

int CSE [6] = {100, 11, 22, 33, 55, 61}

Array ↑ index

100	CSE [0]
11	CSE [1]
22	CSE [2]
33	CSE [3]
55	CSE [4]
61	CSE [5]

CSE [2] + 10;
32

CSE [2] + CSE [5] = 22 + 61
= 83

Types of array : 1) 1D , 2) 2D , 3) Multi dimentional.

Precedence: The priorityx order of operators called precedence

Associativity: When operators has have same ~~associativity~~ precedence then the direction in which the operators are ~~eq~~ evaluated is associativity.

Scanned with CamScanner

Subject CSE1121

Date 09 04 19

1. Name and describe four basic datatypes in C.

Ans: There are mainly ~~four~~ four basic datatype in C. These are-

a) integer
b) Floating point
c) void types
d) character.

a) Integer type: These are whole numbers with a range of values supported by a particular machine.

Formatted Output:

```
int x=123;
printf("%d", x);
```

123

Subject CSE

Date 05 04 19

General form

printf ("control string", arg1, arg2);

%wd / %wf / %wc / %ws

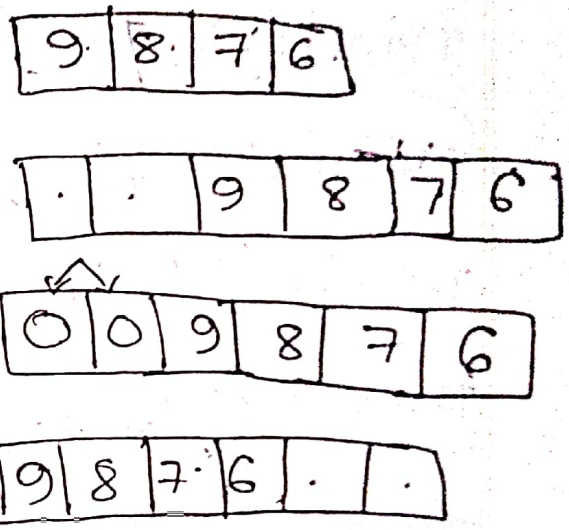
w = w specify minimum width for the output

printf ("%d", 9876)

printf ("%6d", 9876) R.J

printf ("%06d", 9876)

printf ("%-6d", 9876) L.J



int m = 12345

long n = 987654

printf ("%d\n", m)

printf ("%10d\n", m);

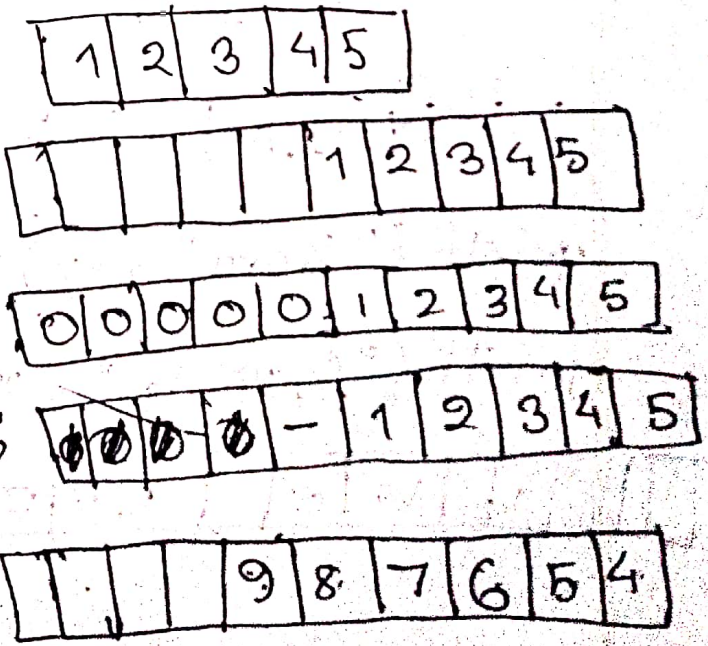
printf ("%010d\n", m);

printf ("%0#10d\n", -m);

printf ("%10ld", n);

w.pf %7.4

98.7341

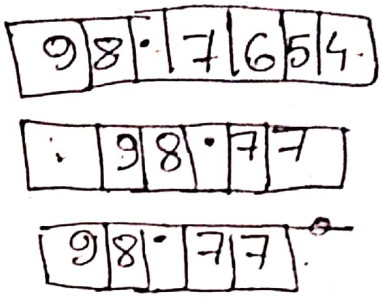


Scanned with CamScanner

Subject

Date

x = 98.7654



~~printf("%f", x);~~

~~printf("%4.2f~~

printf("%7.4f", x);

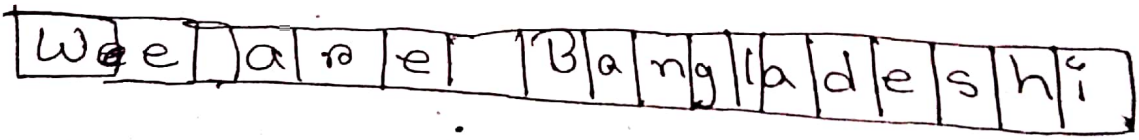
~~printf("%7~~

printf("%7.4f\n", x);

printf("%7.2f\n", x);

printf("%-7.2f\n", x);

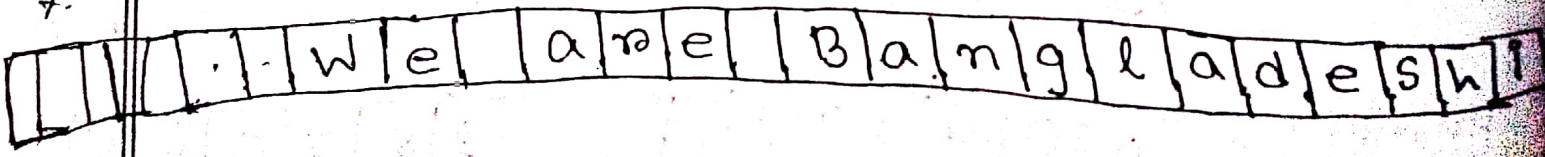
~~Space~~ string xy = "We are Bangladeshi



printf("%s\n", xy);

printf("%25s\n", xy);

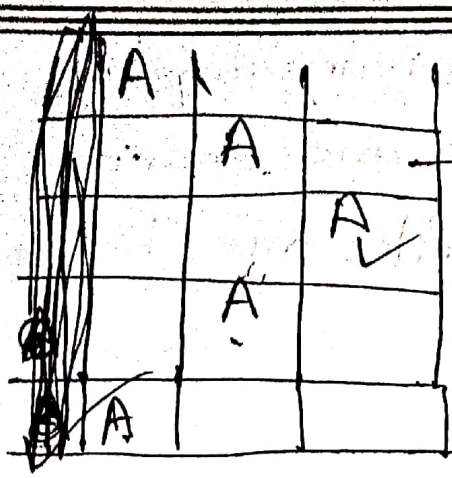
7.



Scanned with CamScanner

Subject

Date



printf("%c", A);

→ printf("%2c", A);

printf("%3c", A);

printf("%-2c", A);

printf("%c\n", A)

A

A= Sobuj khan

10

Sobuj Khan

Sobuj Khan

Sobuj

printf("%10s\n", A);

printf("%16s\n", A);

printf("%-10.5s\n", A);

Scanned with CamScanner

header file: In programming library functions are ~~grop~~ grouped category-wise and stored in different files. These files are known as header files.

*Difference betw "while & do-while".

While	Do-while
1. Condition is at top	1. Condition is at the bottom.
2. No ~~necessa~~ necessity of bracket if there is single statement in body	2. Brackets are compulsory even if there is a single statement.
3. There is no semi-colon at the end of while	3. The semicolon is compulsory at the end of do-while
4. Computer executes the body if and only if condition is ~~not~~ true.	4. Computer executes the body at least once even if condition is false
5. This should be used when condition is more important	5. This should be used when the process is important.

Scanned with CamScanner

Subject | Date

While	Do-while
6. This loop is also refered as entry controlled loop.	6, This loop is also refered as exit controlled loop.

7. while (n<10)
{
print("u%d\n", n);
n++
}
	8. Do
	{
	printf("u%d\n", n);
	} while (n<=100);

Purpose of switch statement.

1) Complexity of the program increases as the number of constants to compare increases. Switch statement provides a way to write easy to manage and readable code.

2) Switch statment provides default case handler when no case matches.

3) We can nest switch statement like if else statement.

Scanned with CamScanner

Subject

Date

	break	Continue
1	A break can appear in both switch & loop (for, while, do) statement.	A continue can appear in only in loop (for, while, do) statements.
2	Used to terminate the loop or to exit loop from a switch	Used to trasfer the control to the start of loop
1.	Break is used to break loop or iteration	Continue continues the loop or iteration
2	Used in swich case	Not used in switch case
3	keyword used "break"	Keyword used is "continue"
4	Breaks loop & comes out from it	Allows iterating in the loop.
5	Control is transfered outside the loop	Control remain in the same loop.

Scanned with CamScanner

Subject

Date

goto use:

```
#include<stdio.h>
int main()
{ int i,sum=0;
 for(int i=0; i<=10; i++){
    {
     sum=sum+i;
     if (i==5){
          go to addition;
        }
     }
  addition:
  printf("%d",sum);
 return 0;
}
```

High level language

1. H. l. l. are programmer friendly. They are easy to write, debug & maintain.
2. It provide higher level of abstraction for machine language
3. It is machine independent language
4. Easy to learn
5. Less error prone, easy to find & debug error
6. Result in better programming product -ivity.

Scanned with CamScanner

5. Show error at last

5. Read & check line by line and show error line by line.

Subject

Date

Compiler

1. Takes ~~Entire~~ Program as ~~input~~ single input and converts it into Object code which is stored.

2) Compiler compile the program as whole

2) Generate object file

3) compiler translation is fast

4) Will stop translating if it finds an error and there will be no executable code generated

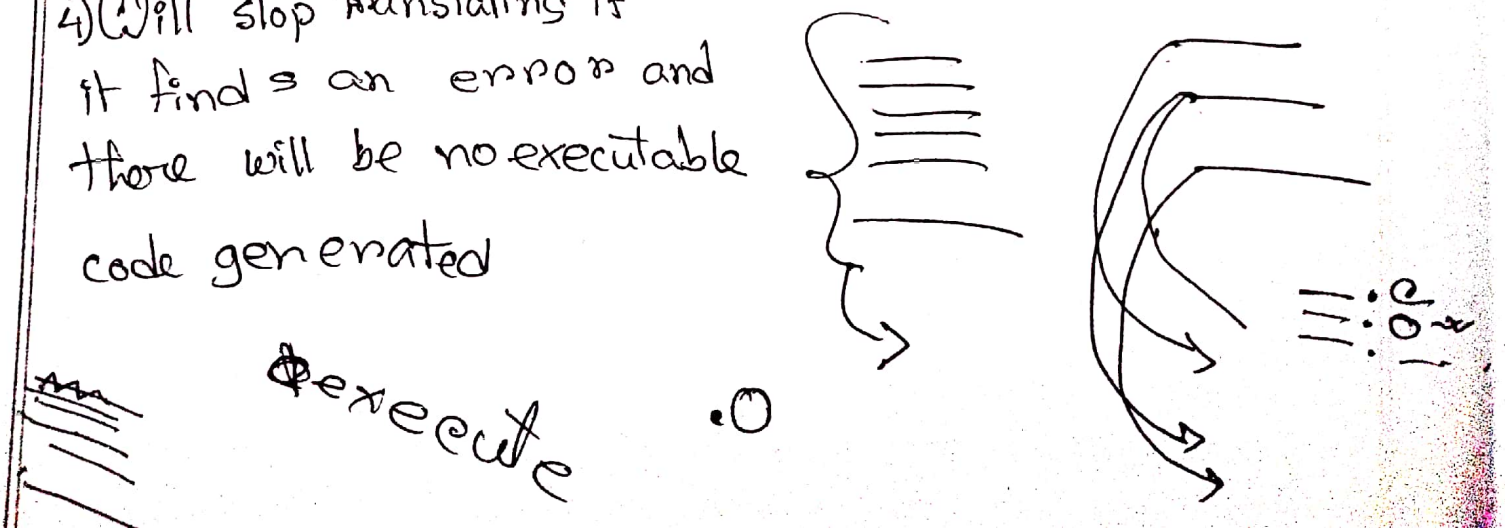
Interpreter

4. ~~Takes~~ ~~Reads~~ Interpreter will execute all the line before error and will stop at the line which contains the error

1) It converts one statement of a program ~~at~~ one at a time

2) ~~It~~ Generates no object file.

3) Interpreter translation is slow.



Scanned with CamScanner

Subject | Date

Algorithm

1. Well defined sequence of steps that provides a soln for a given problem
2. Written in natural language
3. Helps to simplify and understand the problem
4. Example
   a. Get the hourly pay rate
   b. Get the no number of hours worked.

Pseudo Code

Is one of the methods that can be used to represent algorithm.

- In a format that is closely related to high level programing language structure
- Does not use specific prog lan syntax.
- A method of developing an algorithm

a. Input hours
b. Display "Enter the number of hours"

Variables
→ a variable is a container (storage area) to hold data.
To indicate the storage area, each variable be given a unique name (or an identifier).

Constants is a value (or an identifier) whose value cannot be altered in a program Example: 1, 2.5, 'C'

Scanned with CamScanner

Subject CSE 1121

Date 24 04 '19

Array declaration

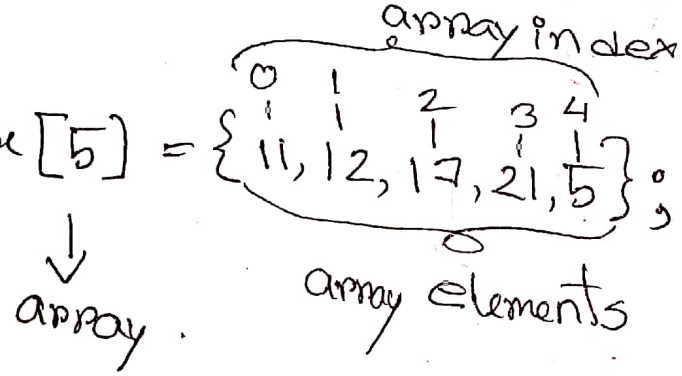
syntax / General form;

Datatype arrayname[arraysize];

int x[5];

float y[100];

CH-1 to 6 all the Q. write



~~int i;~~ for(i=0; i<5; i++)
~~int x[5];~~ printf("%d\n", x[i]);

for(i=0; i<5; i++)
scanf("%d\n", &x[i]);

Scanned with CamScanner

Subject CSE1121 | Date 25 | 04 | 19

1. Get input & print ~~out~~ those array ~~in~~ elements
2. Summation of array elements
3. Make calculator using array

27.04.19

Array

Two-dimentional array:

	col 0	col 1	col 2
row 0	5	6	7
row 1	8	9	10
row 2	11	12	13

General form of 2d-array

datatype array_variable name [row size] [col. size];

	0	1	2	3
0	72	5	11	2
1	1	2	6	9
2	71	62	61	5

int x[3][4] = {72, 5, 11, 2, 1, 2, 6, 9, 71, 62, 61, 5}

= { {72, 5, 11, 2},
{1, 2, 6, 9},
{72, 5, 71, 2} }

[0,0]	[0,1]	[0,2]	[0,3]
[1,0]	[1,1]	[1,2]	[1,3]
[2,0]	[2,1]	[2,2]	[2,3]

int x[0][0] = 72 x[1][2] = 6
x[0][1] = 5 x[1][3] = 9
x[0][2] = 11 x[2][0] = 71
x[0][3] = 2 x[2][1] = 62
x[1][0] = 1 x[2][2] = 61
x[1][1] = 2 x[2,3] = 5

Scanned with CamScanner

Subject

Date

pc pc
mn pq
n!=p → no output

$$A = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \quad B = \begin{bmatrix} 7 & 9 \\ 11 & 15 \end{bmatrix}$$

i=1
j=1,2
k=

$$C = A \times B = \begin{bmatrix} 1\times7 + 2\times11 & 1\times9 + 2\times15 \\ 5\times7 + 6\times11 & 5\times9 + 6\times15 \end{bmatrix}$$

$$= \begin{bmatrix} 29 & 39 \\ 101 & 135 \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}\cdot B_{11} + A_{12}\cdot B_{21} & A_{11}\cdot B_{12} + A_{12}\cdot B_{22} \\ A_{21}\cdot B_{11} + A_{22}\cdot B_{21} & A_{21}\cdot B_{12} + A_{22}\cdot B_{22} \end{bmatrix}$$

```
int i,j,k;
int A[2][2] = {1, 2, 5, 6};
int B[2][2] = {7, 9, 11, 15};
int c[2][2];
for (i=1; i<=2; i++)        → Row
  { for (j=1; j<=2; j++)    → column
      {
       for (k=1, k<=2; k++)
          sum = sum + A[i][j] * B[i][k]
      }
  multiply[i][j] = sum;
   sum=0;
  }
}
```

Scanned with CamScanner

Subject

Date

Assingmant
(i) Multi-dimentional array.
(ii) dynamic array
(iii) Matrix multiplication.

```
printf("Product of the two matrices:\n");
for (i=0; i<=2 ; i++)
  { for (j=0; j<=2 ;j++)
        printf("%d\t", multiply[i][j]);
    printf("\n");
  }
return 0;
for(i=1 i<=2 ; i++)
    for (j = 1; j <= 2; j++)
        for (k=1; j<=2; k++
```

Scanned with CamScanner

Global.	local.
1. Visible in all func	Visible only within the func they are defined
2. Zero by default	2. Uninitialized.
3. A change made by the func is visible everywhere in the Program	3. Any changes made on the value are lost when the func terminates.

4.

b = ~~a~~ ~~b~~ % 5 % (5 - 5 / 2) * (5 - 3) + 5 ;

5 % (5 - 2) * (2) + 5

5 % 2 * 2 + 5

~~10 * 5~~ 5 %

7

2) 5 (2

4

1

2 * 2 + 5

4 + 5

9

Subject CSE1121 | Date 29 04 19

CH-9 (User defined Function)

Function: A group of statements that together perform a task. Every C prog. has at least one func which is main(), A large program in C can be divided to many sub-program. The subprogram posses a self contain components and have well define purpose. The subprogram is called as a func.

```
return_type fun_name(parameter list)
  {
    body of func;
  }

int main ICE( )
  {

  }
```

identifier - variable, func, class.

Types of func

- Library (Built-in) fun.
  - They are written in header files

Scanned with CamScanner

Subject

Date

User defined fun.

Written by user at the time of programming

Advantage of fun.

1. It is much easier to write a structure program where a large program can be divided into a smaller, simpler task.

2. Allowing the code to be called many times

3. Easier to read and update

4. It is easier to debug structured program where there error is easy to find and run.

5. Func can be used in other program.

Elements of func

1. Fun Prototype
2. Fun call
3. Fun argument & parameter
4. Fun Definition

Scanned with CamScanner

Subject | Date

Func Prototype:

- It specify the type of value that is to be return from the func and that is to be passed to the func.
- It is defined in the begining before the func call is made.

☐ Syntax:
☐ return-type name of func (list of argument)

Void sum(int, int).

Func Call: (using

- A func can be called by specifying name and list of argument enclosed in paran thesis & seperated by comma.

```
#include<stdio.h>
int show()
{ printf("Its a func");
}
int main()
{ show();
}
```

Subject | Date

Func arguments & Parameters

- Arguments are also called actual parameters.

```
main()
{
 add (10,20) //Argument
}
add (.)
```

```
#include <stdio.h>
#include<conio.h>
void sum(int a, int b); // func prototype
void main ()
{
int a,b;
printf("Enter two integers\n");
scanf ("%d %d",&a,&b);
 sum (a,b); // func call.
getch ( );
}
void sum (int a,int b)//func definition
{
printf ("Sum %d", a+b);
}
```

actual argument

formal parameter

Scanned with CamScanner

Subject | Date

Fun Return Statement

- It is the last statement of fun that return certain value.
- It return certain types of values to the place from where the func. was invoked.

Syntax:

return (variable-name or constant);

```
int Add(int x, int y)
{ int z;
  z = x+y;
  return z;
}
int main()
{ int a;
  a = Add(5,11);
  printf("%d", a);
```

Func Categories

- void Add() • no argument and no return
- void Add(int x) • with arguments and but no return
- int Add() • No argument & return
- int Add(int x, int y) • Func with argum & return,

Scanned with CamScanner

Local Variable: Variables that are declared inside a func or block are l.v. They can be used only by statements that are inside that func or block of code. Local variables are not known to the ~~funcs~~ ~~out~~ outside

Global : Are defined outside da func., usually at top of the prog. They will hold the value throughout lifetime of your program & can be accessed ~~wit~~ inside any of func defined for program.

Method of calling func.

Call by Value	Call by reference
A copy of data is made and stored by way of name of parameter.	1. Reference (address)

Scanned with CamScanner

Subject

Date

Any change to the parameter have NO affect on data in the calling func.

```
void c(int n, int m)
{ int temp
  n = m,   m = temp;
}

int main(){
   int n=50, m=70
  c(n, m);
```

```
void swap(&a, &b);
{
void swap(int*x, int*y)
{ int t;
  t = *x
  *x = *y
  *y = t;
}
```

Scanned with CamScanner

Subject

Date

```
#include <stdio.h>
#include <conio.h>
void swap(int *, int *);
void main()
{int a,b;
 clrscr();
 printf("Enter the two integers\n");
  scanf("%d%d",&a,&b);
 printf("a=%d and b=%d before calling fun\n",a,b);
 swap(&a,&b);
 printf("a=%d & b=%d after calling fun\n",a,b);
 getch();
 }
void swap(int *a,int *b)
{ int temp;
   temp = *a;
   *a = *b;
   *b = temp;
}
```

pointer → to locate memory address of variables.

Scanned with CamScanner

Subject ARRAY of func

Date

```
#include <stdio.h>
float average (float a[]);
int main(){
float avg, c[] = {23.4, 55, 77.6, 3.405, 1};
avg = average (c);
printf ("Avg age = = %.2f", avg);
return 0;
}
float average (float a[]) {
 int i;
 float avg, sum = 0.0;
for (i=0; i<6; ++i) {
  sum += a[i];
 avg = (sum/6);
 return avg;
}
```

Scanned with CamScanner