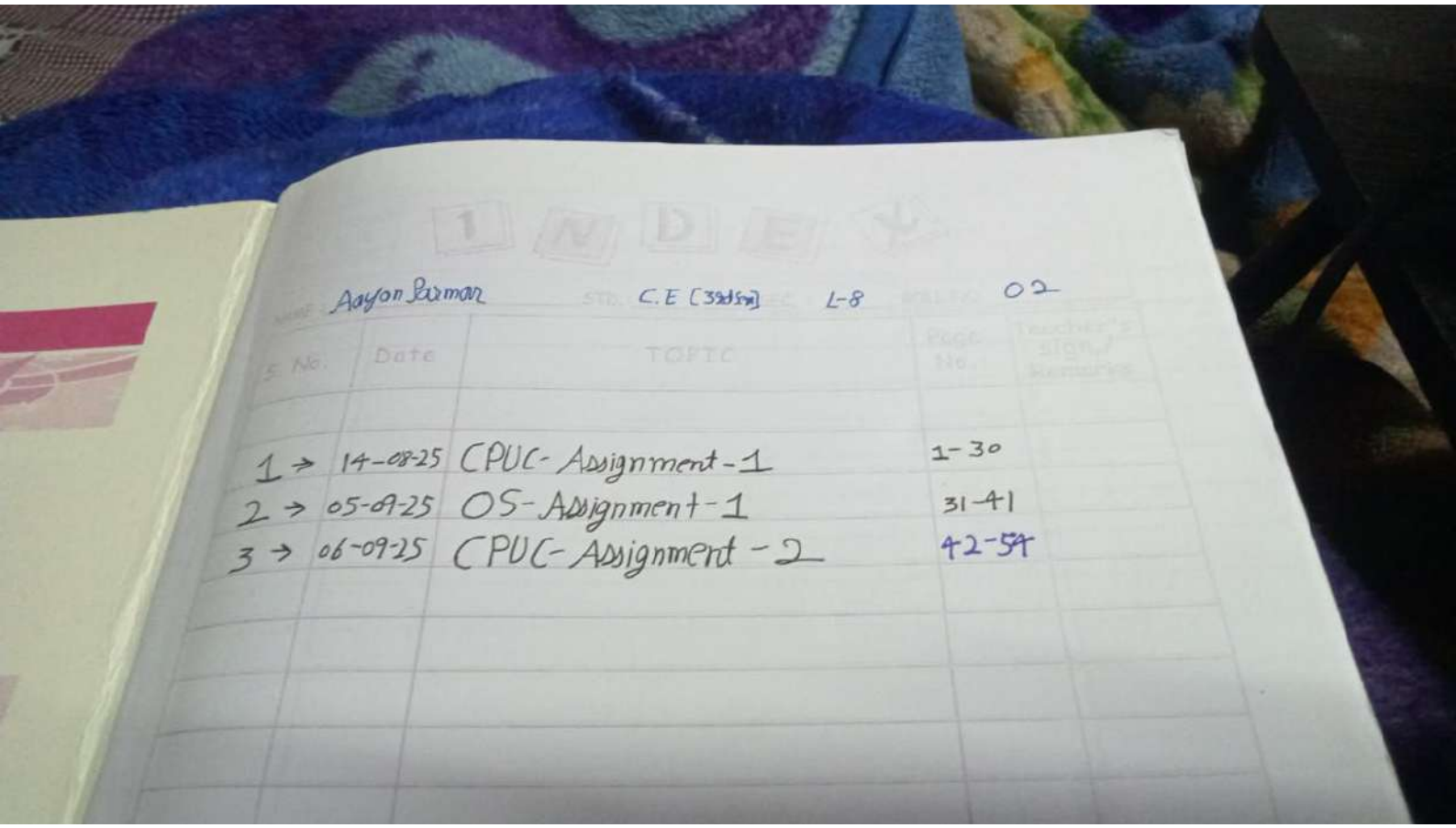
INDEX
Name Aayon Barman STD. C.E [3rd Sem] L-8 02
S. No. Date TOPIC Page No. Teacher's sign/ Remarks
1 → 14-08-25 CPUC - Assignment - 1 1-30
2 → 05-09-25 OS - Assignment - 1 31-41
3 → 06-09-25 CPUC - Assignment - 2 42-54

CPUC

Assignment - 2

Unit: 3 & 4

• Date of Submission :- 09/09/2025

Q1 Explain in brief, the following I/O functions with suitable examples.

☆→ Formate I/O functions :-

• printf() :- It is used to display output on the screen.

Syntax :- printf("Format", var1, var2, ...);

example :-
```
int age = 25;
printf(" I am %d years old.\n", age);
```

Explanation :- %d is a format specifier for an integer.

age is the variable whose value will be printed in place of %d

scanf() :- It is used to take input from the user through the keyboard.

Syntax :- scanf("Format", &var1, &var2, ...);

Example :- int age;
scanf("%d", &age);
printf("You entered: %d \n", age);

Explanation :- %d is the format specifier for an integer.

&age passes the address of the var to store user input.

Unformate I/O Functions :-

1→ getch() :- It reads a single character from the keyboard without displaying it on the screen (no echo).

Header file :- #include ~~<stdio.h>~~ <conio.h>

Example :-
```
#include <conio.h>
#include <stdio.h>

int main()  {
    char ch;
    printf("Press any key: ");
    ch = getch();
```

printf(" You pressed : %c ", ch);
return 0;
}

2→ getche() :- It reads a single character from the keyboard and displays it (with echo).

Header file :- #include <conio.h>

Example :-
```
#include <conio.h>
#include <stdio.h>
int main()
{
  char ch;
  printf("Press any key : ");
  ch = getche();
  printf(" You pressed : %c", ch);
  return 0;
}
```

3→ getchar() :- It reads a single character from standard input. It waits for Enter key.

Header file :- #include <stdio.h>

Now,

Example:-
```
#include <stdio.h>
int main() {
    char ch;
    printf("Enter a character: ");
    ch = getchar();
    printf("You entered: %c", ch);
    return 0;
}
```

4→ putchar():- It displays a single character on the screen.

Header file:- #include <stdio.h>

Example:-
```
#include <stdio.h>
int main() {
    char ch = 'A';
    putchar(ch);
    return 0;
}
```

5→ gets():- It reads a string from the keyboard [until ENTER is pressed].

Note:- gets() is unsafe and deprecated in modern C standards.

Header file:- #include <stdio.h>

Example:-
```
#include <stdio.h>
int main() {
    char name[50];
    printf("Enter your name: ");
    gets(name);
    printf("Hello, %s", name);
    return 0;
}
```

6→ puts():- It displays a string on the screen [automatically adds a newline at the end.].

Header file :- #include <stdio.h>

Example:-
```
#include <stdio.h>
int main() {
    char name[] = "Alice";
    puts(name);
    return 0;
}
```

Q2 What are formate specifiers used in C language.

Ans:- Format Specifiers in C are used with functions like printf() and scanf() to indicate the type of data being input output.

- They act as placeholders for value of variable

ASHOKA
Date
Page No. 47

Common Format Specifiers

	Specifier	Description	Data Type
1→	%d	Integer (decimal)	int
2→	%i	Integer (Same as %d)	int
3→	%f	Floating Point	float
4→	%lf	Double Precision Float	double
5→	%c	Single Character	char
6→	%s	String	char[]
7→	%u	unsigned int	unsigned int
8→	%ld	long int	long
9→	%lld	long long int	long long
10→	%%	print % symbol	—

Example:-

```
#include <stdio.h>
int main() {
    int i = 10;
    float f = 3.14;
    char c = 'A';
    char str[] = "Hello";

    printf("Integer: %d \n", i);
    printf("Float: %.2f \n", f);
    printf("Character: %c \n", c);
    printf("String: %s \n", str);

    return 0;
}
```

- **Notes:-**

1→ Always match the format specifier with the variable type.

2→ Mismatch can cause unexpected behavior or errors.

3→ For input with scanf(), always use the address operator (&) for non-array variables.

Q3 Explain with suitable example, all the operators used in C language and also explain the associativity and order of precedence of operators.

Ans:- Types of operators in C language :-

1. Arithmetic operators :- Used for mathematical operations.

op	Description	Example
+	Addition	a + b
-	Subtraction	a - b
*	multiplication	a * b
/	Division	a / b
%	Modulus (remainder)	a % b

2. Relational operators :- Used to compare values.

OP	Description	example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater or equal	a >= b
<=	less or equal	a <= b

3. Logical Operators :- Used to combine multiple conditions.

OP	Description	Example
&&	Logical AND	a>5 && b<10
!	Logical NOT	!(a>5)

4. Assignment Operators :- Used to assign values to the variables.

OP	Description	Example
=	Assignment	a = 5
+=	Add and assign	a += 3
-=	Subtract & assign	a -= 2
*=	Multiply & assign	a *= 4
/=	Divide & assign	a /= 2
%=	Modulo and assign	a %= 2

5. Unary Operators :- Operate on a single operand.

OP	description	Example
++	increment	++a, a++
--	Decrement	--a, a--
-	Unary minus	-a
+	Unary plus	+a
!	Logical NOT	!a

6.

6. Bitwise Operators :- Used to perform operations at the bit level.

OP	Description	Example
&	Bitwise AND	a & b
|		bitwise OR
^	Bitwise XOR	a ^ b
~	Bitwise NOT	~a
<<	left Shift	a << 1
>>	Right Shift	a >> 1

7. Conditional Operator :- It returns one of 2 values. depending on a condition

OP	Description	example
?:	Ternary Operator	a > b ? a : b

8→ Comma Operator :- It allows multiple expressions to be evaluated in a single statement.

OP	Description	Example
,	Separates expressions	a = (b=3, b+2)

9→ Sizeof Operator :- It returns the size in bytes of variable or data type. It is evaluated at compile time, not at runtime.

OP	Description	example
sizeof	Returns size of a data/var type	sizeof(int)

A→ Operator Precedence and Associativity table

- Operator precedence defines which operators are evaluated first in expressions.
- Associativity defines the direction of evaluation. (left to right or right to left).

Q→ Precedence Table [Highest to lowest]:-

Precedence	Operators	Description	Associativity
1	() [] . →	Function call, array, struct access	left to right
2	++ -- + - ! ~ sizeof	Unary OP.	Right to left
3	* / %	Multiply / Div / Modulus	left to right
4	. + -	Add / Sub	left to right
5	<< >>	Bitwise Shifts	left to right
6	< <= >	Relational	left to right
7	== !=	Equality	left to right
8	&	Bitwise AND	left to right
9	^	Bitwise XOR	left to right
10	|	|	Bitwise OR
11	&&	Logical AND	left to right
12	|	|	|

13	?:	Ternary	Right to left
14	= += -= etc	Assignment	Right to left
15	,	Comma	Left to right

Example:-

```
#include <stdio.h>
int main() {
    int a = 10, b = 5, c = 2
    int result;

    result = a + b * c;
    printf("Result : %d \n", result);

    result = (a+b) * c;
    printf("Result : %d \n", result);

    return 0;
}
```

Summary:-

Precedence determines which operator executes first in complex expression.

Operators are grouped by function: arithmetic, logical, relational, etc.

END