Assignment-2 Date: 22.09.2025

Ques-1 Differentiate assignment and initialization.

Answer: Initialization –

Definition – Giving a variable its first value at the time of declaration.

When it happens – At the moment variable is created.

Example: int x = 10; // initialization

→ Here, x is both declared and given an initial value (10) in one step.

Assignment –

Definition – Giving a value to a variable that has already been declared (can be first value or changing later).

When it happens – After the variable exists in memory.

Example:
```
int x;  // declaration
x = 10; // assignment
x = 20; // reassignment (changing value)
```

→ Here, variable x is already exists; we are just assigning or updating its value.

Ques. 2 Write an algorithm for leap year.

Solution- Algorithm: Check Leap year

1. Start
2. Input the year (say Y).
3. Check the following conditions:
   → if Y is divisible by 400, then it is a leap year.
   → if else Y is divisible by 100, then it is not.
   → else if Y is divisible by 4, then it is leap year.
   → Else, it is not a leap year.
4. Output the result.
5. Stop.

Ques. 3 What will be the output of the program?

```
main() {
  int x;
  x = -3 * -4 % -6 / -5;
  printf ("x = %d", x);
}
```

Solution- first of all the processing part;

$x = -3 * -4 \ \% \ -6 \ / -5;$

→ all operators have same precedence - *, %, /
→ It would be evaluated from left to right.

1. $-3 * -4 = 12$
2. $12 \ \% -6 = 0$
3. $0 / -5 = 0$

So, $x = 0$. Answer

Option, (ii) $x = 0$ is correct.

Ques.4 - 1. Difference between ++a and a++.

Solution- Both are increment operators in C, but they differ in when the increment happens.

→ ++a (Pre-increment)
First increases value of a by 1.
Then uses updated value.

Example: int a = 5;
int b = ++a; // a = 6, b = 6

→ a++ (Post-increment)
First uses the value of a
Then increases it by 1.

Example: int a = 5;
int b = a++; // a = 6, b = 5

2. Operators in C

Solution- (i) Bitwise Operators

Used for Manipulating bits of integers.

& (AND)
| (OR)
^ (XOR)
~ (NOT/Complement)
<< (left shift)
>> (right shift)

Example:
```
int a=5, b=3; // binary a = 0101,
              // b= 0011
printf("%d\n", a&b) //1 (0001)
printf("%d\n", a|b); //7 (0111)
printf("%d\n", a^b); //6 (0110)
printf("%d\n", a<<1); //10 (1010)
printf("%d\n", a>>1); //2 (0010)
```

(ii) Increment and Decrement Operators

++a → Pre-increment (increase first, then use)
a++ → Post-increment (use first, then increase)
a-- → Post-decrement (use first, then decrease)
--a → Post Pre-decrement (decrease first, then use)

Example: int a = 10;
printf("%d\n", ++a); // 11
printf("%d\n", a--); // print a 11, then a = 10

(iii) Logical Operators

Used for Logical (Boolean) conditions or operations.

&& (Logical AND) → true if both true
|| (Logical OR) → true if at least one true
! (Logical NOT) → inverts truth value

Example: int x = 5, y = 10;
if (x > 0 && y > 0) // true && true → true
printf("Both +ve\n");
if (x > 0 || y < 0) // true || false → true
printf("At least one true condition\n");
if (!(x > y)) // !(false) → true
printf("x is not greater than y\n");

Ques-5 Discuss the switch case statement in C with example

Solution- Switch Case in C -

→ The switch statement is used to choose one option from multiple based one a variable's value.
→ It's often cleaner than writing many if-else statements.

Syntax-

```
switch (expression) {
    case value1:
        // Code
        break;
    case value2:
        // Code
        break;
    default:
        // code if no case matches
}
```

Example:

```
int day = 3;

switch (day) {
    case 1: printf ("Monday"); break;
    case 2: printf ("Tuesday"); break;
    case 3: printf ("Wednesday"); break;
    default: printf ("Invalid day");
}
```

Output: Wednesday

Key point: Use break to stop execution after a case; otherwise, it "falls through" to the next case.

Ques. 6 Explain String? Write a Program to print Armstrong number?

Solution- String in C -

→ A string is a sequence of characters stored in a character array ending with a null character (\0).

Example: char str[20] = "Hello";

→ 'str' holds character H, e, l, l, o, \0.

Armstrong Number -

→ An Armstrong number of 'n' digits is a number equal to the sum of its digits each raised to the power 'n'.

Example: 153 = 1^3 + 5^3 + 3^3 → Armstrong

Code:

```
#include <stdio.h>
#include <math.h>
int main() {
    int num, temp, sum = 0, digits = 0;

    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;

    // Count digits
    while (temp != 0) {
        digits++;
        temp /= 10;
    }

    temp = num
    // sum of digits raised to power of digits.
    while (temp != 0) {
        int digit = temp % 10;
        sum += pow(digit, digits);
        temp /= 10;
    }

    if (sum == num)
        printf("%d is a armstrong num.\n", num);
    else
        printf("%d is a not an armstrong num.\n",
                num);
    return 0; }
```

Example Output :

Enter a number : 153

153 is an Armstrong number.

Ques.7 Differentiate between getchar and scanf functions? In response to the input statement scanf("%1d%*%d", &year, &code, &count);

the following data is keyed in 19883745.

What value does the computer assign to the variables Year, Code and count?

Solution-

getchar()	scanf()
→ reads one character at a time	→ reads formatted input (int, float, char, string)
→ Reads from buffer directly, no format specifier	→ Reads formatted input, can skip whitespaces.
→ int (ASCII value of character)	→ Depends on format, usually stored in variable

Example:

```
ch = getchar();
```

Example:

```
scanf("%d", &num);
```

Evaluating the 'scanf' statement:

Statement-

scanf("%4d %*%d", &year, &code, &count);

→ Format explanation-

%4d → read first 4 digits as integer → assigned to year.
%* → ingnore next character
%d → read next integer → assigned to code
count → not specified in format, remains uninitalized

→ Input - 19883745

Step by step:

1. %4d → reads 1988 → year = 1988
2. %* → ignore next character → ignores 3
3. %d → reads remaining 745 → code = 745
4. count → not read → value undefined.

Year = 1988
Code = 745
count = ? (undefined)

Ques-8 Explain difference between "if-else" and "switch-case". With example.

Solution-

if - else	Switch case
→ For complex conditions (ranges, relational, logical)	→ For multiple choices of a single variable/value
→ Can test expressions, range, multiple conditions	→ Works only with constant values (like int, char)
→ Gets lengthy if many else-if statements	→ Cleaner for menu-driven / multiple choices problems.
→ Checks conditions sequentially until true	→ Jumps directly to matching 'case'.
→ Slower if many conditions	→ Faster for many fixed values.

Examples:

Using if-else -

```
int x = 3;

if (x == 1)
    printf ("One");
else if (x == 2)
    printf ("Two");
else if printf (x == 3)
    printf ("Three");
else
    printf ("Other");
```

Output - Three

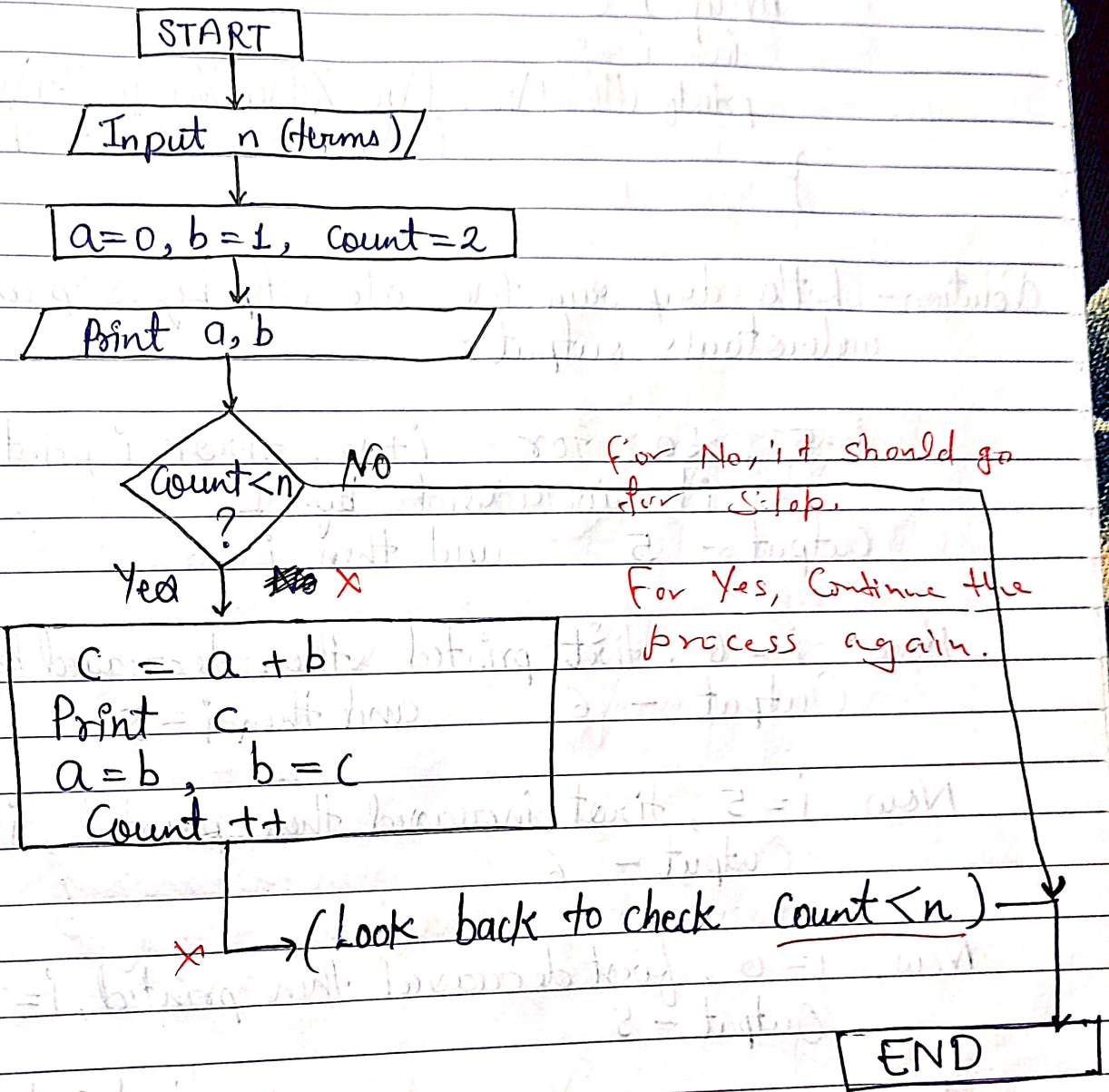
Using switch-case:

```
int x = 3;
switch (x) {
    case 1: printf ("One"); break;
    case 2: printf ("Two"); break;
    case 3: printf ("Three"); break;
    default: printf ("Other");
}
```

Output - Three

Ques. 9 Draw flow chart for generating Fibonacci series

Solution -



Ques-10 Write the output of the following program:

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

Solution - Let's dry run the code step by step and understand output:

i=5, so for i++, first i printed then it increased by 1
Output - 5 and then i=6

Now, i = 6, first printed then decreased by 1,
Output - 6 and then i=5

Now, i = 5, first increased then printed, i=6
Output - 6

Now, i = 6, first decreased then printed, i=5
Output - 5

Last one value of i = 5,
Output - 5

Answer

final Output - 5
6
6
5
5

Ques. 11 Write what is an Operator? Write different types of operators in C.

Solution- An operator is a special symbol that tells the compiler to perform a specific operation on operands (variables/values).

Example: +, −, *, /

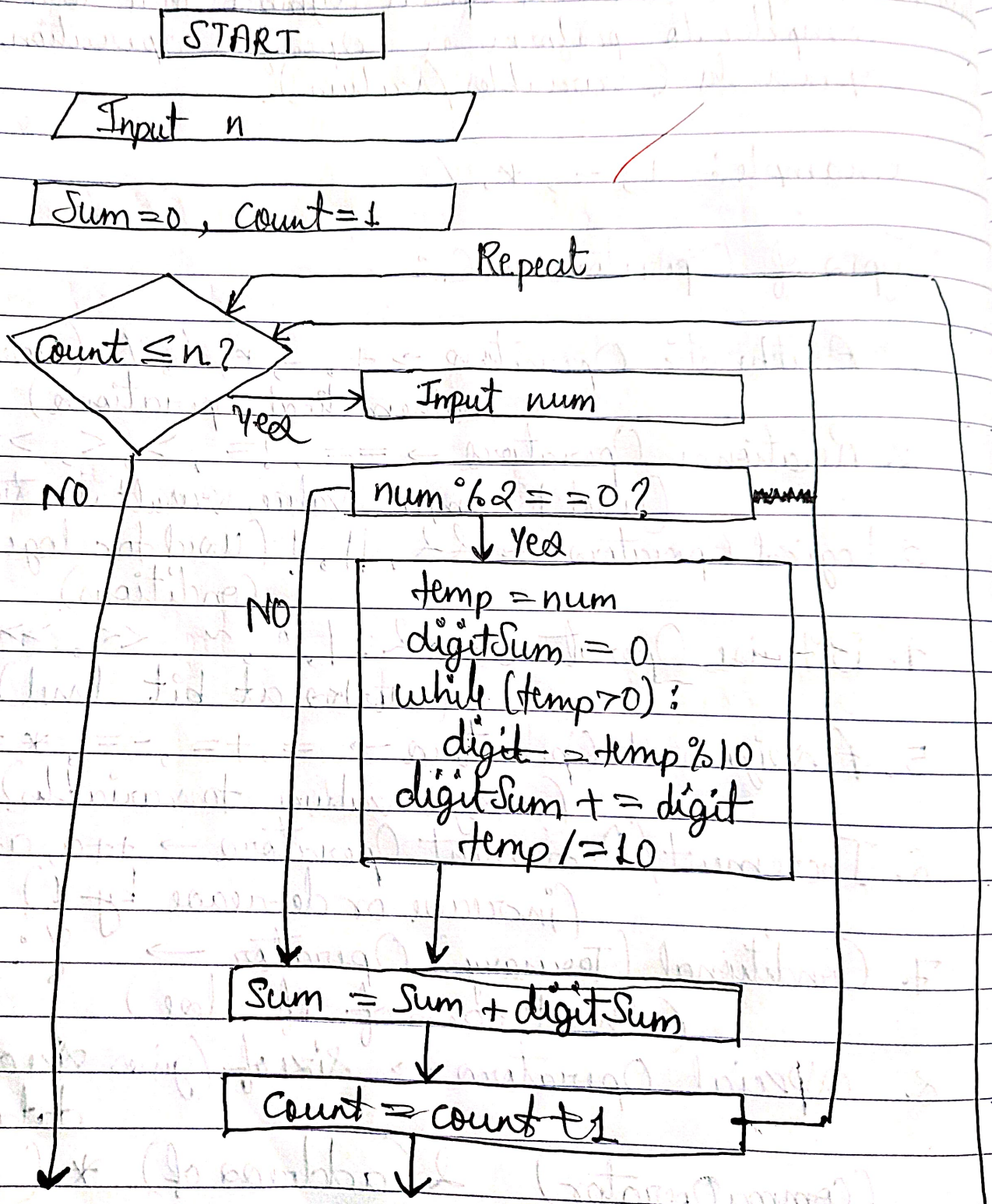
Types of Operators in C:

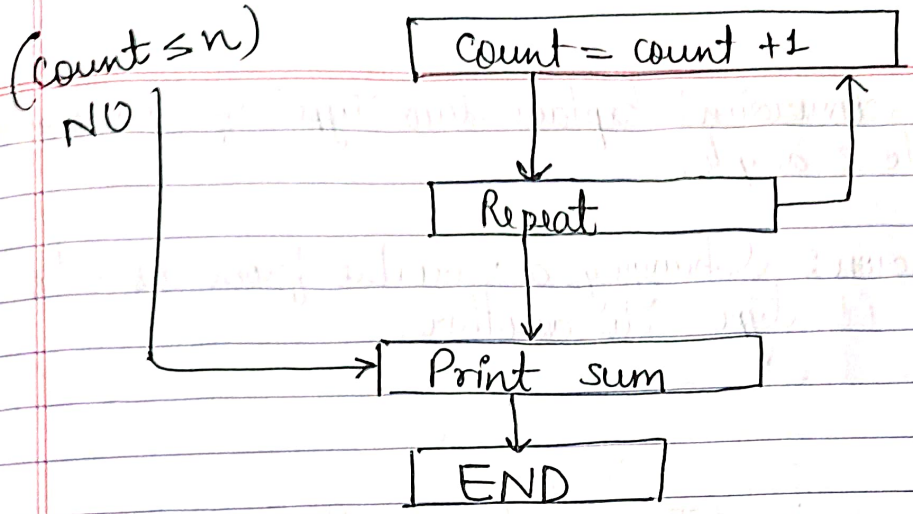
1. Arithmetic Operators → +, −, *, /, % (used for mathematical operations)
2. Relational Operators → ==, !=, >, <, >=, <= (used to compare value, result is true/false).
3. Logical Operators → &&, ||, ! (used for logical conditions)
4. Bitwise Operators → &, |, ^, ~, <<, >> (works at bit level)
5. Assignment Operators → =, +=, -=, *=, /=, %=, (assign values to variables)
6. Increment/Decrement Operators → ++a, a++, --a, a-- (increase or decrease by 1)
7. Conditional (Ternary) Operator → ? : (Short form of if-else)
8. Special Operators → sizeof (gives size of variable/datatype).

, (Comma Operator) & (address of) * (pointer dereference)

Ques. 12 Draw a flow chart to input 'n' numbers from user. Calculate sum of digits of all even numbers and display it.

Solution-





Steps:

1. START
2. Input n (number of values or inputs)
3. Initialize sum = 0, count = 1
4. Loop until count ≤ n:
   - Input number num
   - if number is even:
     - find sum of its digits
     - Add to sum
   - Increment count by 1.
5. After loop ends, display sum.
6. END

Ques.13 What is type conversion? Explain two types of conversion with suitable example.

Solution- Type conversion: Changing a variable from one data type to another.

Types:

1. Implicit Conversion (Type Casting) - Done automatically by the compiler.

Example:
```
int i = 10;
float f = i; // int automatically converted to
             // float.
```

2. Explicit Conversion (Type Casting) - Done manually by the programmer.

Example:
```
float f = 9.5;
int i = (int) f; // float explicitly
                 // converted to int.
```

Ques-14 Explain the following with proper example:
(i) break (ii) continue (iii) goto.

Solution (i) break – Exits a loop or switch immediately.

Example:
```
for(int i=1; i<=5; i++){
    if (i==3)
        break; // loop ends when i is 3.
    printf("%d", i);
}
```

Output: 1 2

(ii) Continue – Skips the current iteration and continues with next.

Example:
```
for(int i=1; i<=5; i++){
    if (i==3)
        continue; // skips printing 3
    printf("%d", i);
}
```

Output: 1 2 4 5

(iii) goto – Jumps to a labeled statement in the program.

Example:

```
int i = 1;
loop:
printf ("%d ", i);
i++;
if (i <= 3)
    goto loop; //jumps back to label 'loop'..

//Output : 1 2 3
```

Ques. 15 What will be the output of the below C programs?

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

Solution- This code will not compile.

Reason:

++a++ is invalid in C.

→ No output will be produced.

Option: c) Compile-time error.

Answer

Ques. 6 What will be the output of the below C program?

```
#include <stdio.h>
int main() {
    int a = 3, b = 9
    printf("%d", ++(a * b+1));
    return 0;
}
```

Solution - This program will not compile.

Reason:

++(a * b+1) is invalid in C.

→ Compiler throwed an error: lvalue required as increment operand.

Option: a) Compile-time error.

Ques-17 What will be the output of the below C program?

```
#include <stdio.h>
int main() {
  int a=2, b=1, c=0;
  c = a +++ b;
  printf ("%d %d %d", a, b, c);
  return 0;
}
```

Solution- The expression a+++b is parsed as a++ +b (because of operator presedence).

So :

- a++ → post increment, value used = 2, then a becomes 3.
- Expression becomes 2+b → 2+1 = 3.
- So, c=3

Final values:

- a = 3
- b = 1
- c = 3

Option: a) 3 1 3 Answer

Ques. 18 What will be the output of the following program in C?

```
#include <stdio.h>
int main() {
  int i=4;
  switch (i) {
    default:
        printf("%s", "Default");
    case 0:
        printf("%s", "Case 0");
    case 1:
        printf("%s", "Case 1");
    case 2:
        printf("%s", "Case 2");
  }
  return 0;
}
```

Solution- In according to code, their is no case 4, so, default will execute and no break statement is there so line by line all cases will execute.

Output: DefaultCase 0Case 1Case 2

Ques-10 (What will be the output of the following program in C?

```
#include <stdio.h>
void int main() {
    char *ch;
    printf("enter a value between 1 to 3: ");
    scanf("%s", ch);
    switch(ch) {
        case "1": printf("1");
                  break;
        case "2": printf("2");
                  break;
    }
}
```

Solution- the code will not execute,

reason: error: switch quantity not a integer

error: case label does not reduce to an integer constant

error: char *ch; is an uninitialized pointer → leads to undefined behaviour when scanf writes to it.

Ques. 20 What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input).

Solution- Code -

```
#Include <stdio.h>
void main() {
    int ch;
    printf ("enter a value between 1 to 2: ");
    scanf ("%d", &ch);

    Switch (ch, ch+1) {
    case 1:
        printf ("1\n");
        break;
    case 2:
        printf ("2\n");
        break;
    }
}
```

The Line:

```
Switch (ch, ch+1)
```

This uses the comma operator:

- Expression (ch, ch+1), evaluates both but returns the value of the last operand → ch

if input = 1:

- ch = 1
- Switch expression = ch + 1 = 2

So control goes to case 2: → prints 2

Output: 2