Expression_evaluation

Damodar Kammili

EXPRESSIONS

```
#include <stdio.h>
int main() {  // The main function where program execution begins
   // Declare and initialize integer variables
   int a = 12, b = 18, c = 7,d = 4, result;
     result = a + a * -b / c % d + c * d;
   // Print the calculated result as an integer
   printf("%d", result);

   return 0;    // Indicate successful program execution
}
```

Output:

38

```
#include <stdio.h>

int main() {
   float a = 6.0;
   printf("%.2f", (9/5)*a + 11);
   return 0;
}
```

Output:

17.00

```
int main( ) {
   int y = 10;    // Declare an integer variable 'y' and initialize it to 10

   // Declare an integer variable 'z' and assign a value to it.
   // The expression (y == 10) is a relational expression.
   // In C, a true relational expression evaluates to 1, and a false one to 0.
   int z = y + (y == 10);

   // Print the value of 'z' followed by a newline character
   printf("%d\n", z);
```

```
    return 0;      // Indicate successful program termination
}
```

Output:

11

```
int main( ) {
  int h = 9;
    // Declare an integer variable 'b' and assign a value based on a ternary operator.
    // The expression before '?' is the condition.
    // If the condition is true, the value before ':' is assigned to 'b'.
    // If the condition is false, the value after ':' is assigned to 'b'.
    int b = 5 * 2 + 2 * 3 < h * 2 ? 5 : 3;

    // Print the value of 'b' as an integer
    printf("%d", b);
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int p, t, si;
    float r;
    p = 5000;
    t = 4;
    r = 7.5;
    si = (p * t * r) / 100;
    printf("%f", si);

    return 0;
}
```

Output:

0.0000000000

```
int main()
{
   int a = 0101;
   printf("\n a=%d", a);
   return 0;
}
```

Output:

a=65

Type Conversion in Assignments

—----------

```
int i ;
 float b ;
 i = 3.5 ;
 b = 30 ;
```

Here in the first assignment statement though the expression's value is a float (3.5) it cannot be stored in ***i*** since it is an int datatype. In such a case the float is demoted to an int type and then its value is stored. Hence what gets stored in ***i*** is 3. Exactly opposite happens in the next statement. Here, 30 is promoted to 30.000000 and then stored in ***b***, since ***b*** being a float variable cannot hold anything except a float value.

Example 2: It has been assumed that k is an integer variable and a is a real variable

Arithmetic Instruction	Result	Arithmetic Instruction	Result
k = 2 / 9	0	a = 2 / 9	0.0
k = 2.0 / 9	0	a = 2.0 / 9	0.2222

k will get **int** value at the end

a will get **float** value at the end

—----------

Damodar Kammili

Hierarchy of Operators

Priority	Operators	Description
1st	* / %	multiplication, division, modular division
2nd	+ -	addition, subtraction
3rd	=	assignment

—----------

*** Determine the hierarchy of operations and evaluate the following expression:**

i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

Solution :

Stepwise evaluation of this expression is shown below:

i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

i = 6 / 4 + 4 / 4 + 8 - 2 + 5 / 8 operation: *

i = 1 + 4 / 4 + 8 - 2 + 5 / 8 operation: /

i = 1 + 1+ 8 - 2 + 5 / 8 operation: /

i = 1 + 1 + 8 - 2 + 0 operation: /

i = 2 + 8 - 2 + 0 operation: +

i = 10 - 2 + 0 operation: +

i = 8 + 0 operation : -

i = 8 operation: +

—----------

What is the output of the following expression after evaluation?

kk = 3 / 2 * 4 + 3 / 8 + 3

Solution :

7

—----------

Associativity of Operators

—----------

—----------

—----------

C programs are converted into machine language with the help of

(1) An Editor

(2) A compiler

(3) An operating system

(4) None of the above

Output:

2

—----------

What is the output of the following program

```
#include<stdio.h>

void main( )
{
int a, b ;
a = -3 - - 3 ;
b = -3 - - ( - 3 ) ;
printf ( "a = %d b = %d", a, b ) ;
}
```

Output:

a = 0 b = -6

—----------

What would happen when you try to execute the program?

```
void main( )
{
float a = 5, b = 2 ;
int c ;
c = a % b ;
printf ( "%d", c ) ;
}
```

Output:

Compilation error

Reason(Since modulus operator does not works with float type values)

—----------

In b = 6.6 / a + 2 * n ; which operation will be performed first?

(1) 6.6 / a

(2) a + 2

(3) 2 * n

(4) Depends upon compiler

Output:

1

—----------

If a is an integer variable, a = 5 / 2 ; will return a value

(1) 2.5

(2) 3

(3) 2

(4) 0

Output:

3

—----------

Which of the following shows the correct hierarchy of arithmetic operations in C

(1) / + * -

(2) * - / +

(3) + - / *

(4) * / + -

Output:

4

—----------

What will be the value of d if d is a float after the operation d = 2 / 7.0?

(1) 0

(2) 0.2857

(3) Cannot be determined

(4) None of the above

—----------

A character variable can never store more than

(1) 32 characters

(2) 8 characters

(3) 254 characters

(4) 1 character

Output:

4

reason:

A `char` variable in C is designed to store a single character. If you want to store more than one character, you need an array of characters (a string).

—----------

The statement char ch = 'M' would store in ch

(1) The character Z

(2) ASCII value of Z

(3) Z along with the single inverted commas

(4) Both (1) and (2)

Output:

2

—----------

Identify the one which is not a binary operator?

(1) &&

(2) ||

(3) * , /

(4) !

Output:

4

—----------

—----------

—----------

—----------

—----------

Damodar Kammili

—----------

—----------

if-else condition programs

Damodar Kammili

```
#include <stdio.h>          //positive or negative
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);                  // Input a number
   // Check if the number is positive or negative
    if (num >= 0) {
        printf("The number is positive or zero.\n");
    } else {
        printf("The number is negative.\n");
    }
    return 0;
}
```

```
int main() {                    //vowel or consonant
    char ch;
    printf("Enter a character: ");
    scanf("%c", &ch);                   // Input a character
    // Check if the character is a vowel or consonant
    if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
        ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
        printf("%c is a vowel.\n", ch);
    } else {
        printf("%c is a consonant.\n", ch);
    }
    return 0;
}
```

```
printf("Enter a number: ")              // even or odd
        scanf("%d", &num);
    if (num % 2 == 0) {
        printf("The number %d is even.\n", num);
    } else {
        printf("The number %d is odd.\n", num)
}
```

output:

```
int main() {                    // Leap Year Checker
  int year;
    printf("Enter a year: ");      // Input a year
    scanf("%d", &year);      // Check for leap year
    if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0))) {
        printf("%d is a leap year.\n", year);
    } else {
        printf("%d is not a leap year.\n", year);
    }
    return 0;
}
```

```
int main() {                    //Grade Checker (Using if-else ladder)
    int marks;

    // Input marks
    printf("Enter marks: ");
    scanf("%d", &marks);
    if (marks >= 90) {                // Check the grade using an if-else
ladder
        printf("Grade: A\n");
    } else if (marks >= 75) {
        printf("Grade: B\n");
    } else if (marks >= 50) {
        printf("Grade: C\n");
    } else if (marks >= 35) {
        printf("Grade: D\n");
    } else {
        printf("Grade: F\n");
    }
    return 0;
}
```

```
    if (age >= 18) {
```

```
        printf("You are eligible to vote.\n");     // Check voting eligibility
    } else {
        printf("You are not eligible to vote.\n");
    }
```

```
int main() {                    //Simple Bank Account Balance Check
    double balance, withdraw;
    printf("Enter your account balance: ");     scanf("%lf", &balance);  1000
    printf("Enter the withdrawal amount: ");   scanf("%lf", &withdraw); 1500
    if (withdraw <= balance) {          // Check if withdrawal is possible
        printf("Withdrawal successful! Your remaining balance is %.2lf\n", balance - withdraw);
    } else {
        printf("Insufficient funds! You cannot withdraw %.2lf.\n", withdraw);
    }
```

```
int main() {                    //Divisibility Checker (Complex condition)
    int num;  // Input a number
    printf("Enter a number: ");
    scanf("%d", &num);  //Check if the number is divisible by both 3 and 5
    if (num % 3 == 0 && num % 5 == 0) {
        printf("The number %d is divisible by both 3 and 5.\n", num);
    } else {
        printf("The number %d is not divisible by both 3 and 5.\n", num);
    }
    return 0;
}
```

```
int main() {                                //Largest of two numbers
    int num1, num2;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
                                    // Check which number is largest
    if (num1 > num2) {
        printf("The largest number is %d.\n", num1);
    } else if (num2 > num1) {
        printf("The largest number is %d.\n", num2);
```

```
    } else {
        printf("Both numbers are equal.\n");
    }
 return 0;
}
```

```
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
  // Check if the number is even or odd using the conditional
operator
(num % 2 == 0) ? printf("The number %d is even.\n", num) : printf("The number %d is odd.\n", num);
    return 0;
}
```

```
printf("Enter a number: ");
    scanf("%d", &num);
  // Check if the number is positive, negative, or zero
    if (num > 0) {
        printf("The number %d is positive.\n", num);
    } else if (num < 0) {
        printf("The number %d is negative.\n", num);
    } else {
        printf("The number is zero.\n");
    }
```

```
int main() {
    int a = 1, b = 0, c = 0;
    if (a > b && b > c || c == 0) {
         printf("Condition is TRUE\n");
    } else {
        printf("Condition is FALSE\n");
    }
```

```
int main() {
   int x = 5, y = 10;
 if ((x = y) > 5) {         // Assignment within the condition
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

```
int main() {
   int a = 5, b = 3;
   if (a && b == 1) {
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

```
int main() {
   int x = 4, y = 2, z = 0;
        0
 if (x | y && z) {
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

Output:

Condition is FALSE

```
int main() {     // confusing ELSE condition
   int a = 5, b = 10;
   if (a > 0)
        if (b < 5)
          printf("Inner condition is TRUE\n");
  else
         printf("Inner condition is FALSE\n");

   return 0;
```

```
}
```

Output:

Inner condition is FALSE

```
int main() {
   int x = 0, y = 10;      1010
              0         0        IF( 1)
   if ( !(x && y || !y)) {
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }  return 0;
}
```

Output:

Condition is TRUE

```
#include<stdio.h>
int main() {
   int i = 10;
   if (--i == 0 && i++ == 1) {
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
   printf("Final value of i: %d\n", i);
   return 0;
}
i=10;
a=2;
```

output:

Condition is TRUE

Final value of i: 2

```
int main() {
   int a = 5, b = 10;
   if ( (a > b ? a : b) < (b > a ? b : a) ) {
           if(   10    <     10 )
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
   return 0;
}
```

Output:

Condition is FALSE

```
 int main()
{ int i=0;                                                    i=0
                                                              i=2
int condition1 = 1


int condition2 = 1;
if (1 ) {
   printf("Condition is TRUE\n");
} else {
   printf("Condition is FALSE\n");
}
```

```
printf("Final value of i : %d\n", i);
return 0;
}
```

Output:

// when i=0

Condition is TRUE

Final value of i : 2

// when i=1

Condition is FALSE

Final value of i : 3

```
int main() {
 int  a=2, b=3 ;
int max1 = (a > b) ? a : a*a;                    max1= 4
                                                  max2= 7
int max2 = (b > a) ? b+max1 : b ;
   printf("%d %d ",max1,max2);
if (max1 < max2) {
   printf("Condition is TRUE\n");
} else {
   printf("Condition is FALSE\n");
}
   printf("%d %d ",max1,max2);
}
```

Output:

4 7

Condition is TRUE

4 7

```
int main() {
   int x = 2147483647;

   if (x + 1 < x) {     // Overflow happens here
      printf("Condition is TRUE\n");
   } else {
```

```
        printf("Condition is FALSE\n");
    }
    return 0;
}
```

output :

Condition is True // for 32 bit systems

Condition is FALSE // for 64 bit systems

```
int main() {
    char ch = 255;
    if (ch > 0) {
        printf("Condition is TRUE\n");
    } else {
        printf("Condition is FALSE\n");
    }
    return 0;
}
```

Output:

Condition is FALSE

Reason:

255 is outside the range of a signed `char`, so it wraps around.

On most systems using two's complement, this happens:

`char ch = 255; → ch becomes -1 ( via two's compliment )`

255 in binary: `11111111`

As a signed char, this is interpreted as `-1`

```
int main() {
   int a = 1, b = 2;
   if (a = b , a == 1) {
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
printf( " %d ", a );
   return 0;
}
```

Output:

Condition is FALSE

2

```
int main() {
   unsigned int a = 1;
   int b = -1;

   if (a < b) {
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
   return 0;
}
```

```
int x = 2147483647;

if (x + 1 < x) {
   printf("Condition is TRUE\n");
} else {
   printf("Condition is FALSE\n");
}
```

Damodar Kammili

```
#include<stdio.h>
int main() {
  if ( !!(x > 5)) {
   printf("True\n");
  } else {
   printf("False\n");
    }
}
```

//Will this condition be true or false for x = 10?

True

```
int main( ) {
if ( 0 ) {
   printf("True\n");
} else {
   printf("False\n");
}
}
```

//What will be the output for x = 10 ?

False

```
 int main( ) {
   int x = 10;
    int y = 5;
  if ( 0 ) {
      printf("True\n");
    } else {
      printf("False\n");
    }
}
```

Damodar Kammili

//Will this condition be true or false?

False

```
int x = 5;
if ((x & 1) && (x >> 1)) {
    printf("True\n");
} else {
    printf("False\n");
}
```

What's the output?

```
int arr[5] = {1, 2, 3, 4, 5};
int *p = arr;
if (p + 1 > p) {
    printf("condition becomes True \n");
} else {
    printf("Condition becomes False\n");
}
```

//Is this condition true or false?

Output:

Condition becomes True

//*3. **Enum Comparison***

```
enum Color { RED, GREEN, BLUE };   // here RED- 0 GREEN - 1  BLUE - 2
enum Color c = GREEN;                    // now c  will get 1
if (c > RED && c < BLUE) {
    printf("True\n");
} else {
    printf("False\n");
}
```

Damodar Kammili

What's the output?

True

//*4. Floating-Point Comparison*

```
int main( ) {
  float x = 0.1 + 0.2;      0.30000000001
if (x == 0.3) {
    printf("True\n");
} else {
    printf("False\n");
}
return 0;
}
```

//Is this condition True or false?

False

//*5. Array Indexing*

```
int arr[5] = {1, 2, 3, 4};
if (arr[2] > arr[4]) {
    printf("True\n");
} else {
    printf("False\n");
}
```

// What's the output?

Output :

True

```
int main() {
    int i;
    if(i=0,2,3) {

        printf(" Do you know NPTEL\n");
    } else {
        printf("Programming on C ");
```

```
    }
    printf("%d\n", i);
    return 0;
}
```

reason:

This uses the **comma operator** (,). The comma operator evaluates its operands from left to right and the **result of the entire expression is the value of the rightmost operand which is 3.**

```
#include <stdio.h>

int main() {
    int x = 0;
    if (x++) {
        printf("true\n");
    } else if (x == 1) {
        printf("false\n");
    }
    return 0;
}
```

Output:

false

```
#include <stdio.h>
int main() {
    int a = 1, b = 2, c = 3;
    if (c > b > a) {
        printf("true");
    } else {
        printf("false");
    }
    return 0;
}
```

Output:

false

```
int main() {
   int a = 1, b = -1, c = 0, d;
        a=2    b=0      c= -1
     d = ++a && ++b || c--;
       d= 0
   if (d) {
      printf("Kolkata \n");
   } else if (1) {
        printf("Delhi \n");
       }
       else {
           printf("Bangalore \n");
           }
   return 0;
}
```

Output:

Delhi

```
int main() {
   int a = 1;
   if (a--)      a->0
         {
         printf("Hyderabad\n");
        }
   if (1) {     a->1
      printf(" Avanthi College \n");
   }

   return 0; }
```

Output:

True

False

```
#include <stdio.h>

int main()
{
   if (1)
      printf("Condition is true.");
   else
      printf("Condition is false.");
   return 0;
}
```

Output:

Condition is true

Little guess programs

---

if (**expression**)
statement ;

Examples:

```
if ( 5 )
    printf ( "yes this will work" ) ;
if ( 1 )
   printf ( "why not this one?" ) ;
if ( -5 )
   printf ( "Hush, even negative also works" ) ;
```

—----------

```
main( ) {
 int i ;
 printf ( "Enter value of i " ) ;
 scanf ( "%d", &i ) ;     i -> 5
 if ( i = 5 )
    printf ( "You entered 5" ) ;
 else
    printf ( "You entered something other than 5" ) ;
}
```

Output:

Enter value of i 100

You entered 5

---

i=0;

What will happen if we write a semicolon at the end of if.

```
if ( i == 5 )
    printf ( "You entered 5" ) ;
return 0; }
```

Output:

Enter Value of i 10000

You entered 5

—----------

```
main( ) {
 float a = 12.52 , b = 12.52 ;
 if ( 1){
   printf ( "a and b are equal" ) ;
}
 printf(" a and b are not equal ");
```

```
}
```

Output:

a and b are equal

—----------

```
#include<stdio.h>

   void main( ) {
 int x = 10, y = 15 ;
 if ( x  = y % 3 )
     printf ( " Don't you think it is Com.Err..." ) ;
}
```

Output:

Nothing will be printed

—----------

```
void main( ) {
char a = 'A' ;
Int var = 10;
 printf ( "%c  %d " ,a , a ) ;
}
```

Output:

A 65

—----------

What values of i, j variables could hold?

```
void main( )
{
 int j = 20, k = 22 ;
 if ( k >= j )
 {
     {
        int p=10;
        k = j ;
        j = k ;

    }
 }
 printf("%d %d",k,j);
}
```

Output:

20 20

```
void main( )
{
 int i = -1, j = 1, k ,l ;
 k = !i && j ;
 l = !i || j ;
 printf ( "%d %d", k, l ) ;
}
```

Output:

0 1

```
#include<stdio.h>
void main( )
{
 int age=20;
 if ( age>=18 )
        printf ( "\nAvanthi" ) ;
 else
       printf ( "\n Students of Avanthi" ) ;
}
```

Output:

College

—----------

Method 3

For this method, we use the bitwise operator, in this case, bitwise and (&) How this works is below

- For any number following operation : (num & 1) will always result
    - 1: If num is odd
    - 0: if num is even

—----------

```
for( —------  ;  —------ ; —----- )
{
Set of Statements ;
}

for( id = 24h.. ; condition  ; updation )
{
college
}

Statement - x;
```

Damodar Kammili

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

Damodar Kammili

for loop

```c
// Calculate Sum of Natural Numbers upto 20
#include <stdio.h>

int main() {
    int n, sum = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        sum += i;
    }

    printf("Sum of natural numbers up to %d is %d\n", n, sum);
    return 0;
}
```

```c
// Count and Display Even Numbers
   Between start and end range
#include <stdio.h>

int main() {
    int start, end;
    printf("Enter the start and end range: ");
    scanf("%d %d", &start, &end);

    printf("Even numbers between %d and %d are:\n", start, end);
    for (int i = start; i <= end; i++) {
        if (i % 2 == 0) {
            printf("%d ", i);
        }
    }

    printf("\n");
    return 0;
}
```

```c
// Multiplication table
#include <stdio.h>

int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);

    for (int i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", num, i, num * i);
    }
return 0;
```

```
}
// Count Vowels in a String
#include <stdio.h>
#include <string.h>
int main() {
    char str[100];
    int count = 0;
    printf("Enter a string: ");
    scanf("%s", str);
 for (int i = 0; i < strlen(str); i++) {
        char ch = str[i];
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch
== 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch
== 'U') {
            count++;
        }
    }
  printf("Number of vowels in the string is %d\n", count);
    return 0;
}
```

```
// Find Largest Element in an Array
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    printf("The largest element is %d\n", max);
    return 0;
}
```

1234 = 1+ 2+ 3+ 4

// Sum of digits of a number

```
#include <stdio.h>

int main() {
    int num, sum = 0, digit;
    printf("Enter a number: ");
    scanf("%d", &num);

 for (    ; num != 0; num /= 10) {
        r = num % 10;
        sum += r;
    }
 printf("Sum of the digits is %d\n", sum);
    return 0;
}
```

1234 --> 4321

/ / **Reverse of a number**

```
#include <stdio.h>

int main() {
    int num, reversed = 0, digit;
    printf("Enter a number: ");
    scanf("%d", &num);

    for (; num != 0; num /= 10) {
        digit = num % 10;
        reversed = reversed * 10 + digit;
    }

    printf("Reversed number is %d\n", reversed);
    return 0;
}
```

// Check if a number is Prime

```
#include <stdio.h>

int main() {
    int num, isPrime = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
if (num <= 1) {
        printf("%d is not a prime number.\n", num);
        return 0;
    }
for (int i = 2; i <= num / 2; i++) {
        if (num % i == 0) {
```

```
            isPrime = 0;
            break;
        }
    }

    if (isPrime) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}
```

```
int main()
{

    for ( ; i <= 9; )
    {
        i++;
        printf("%d ", i);
    }
    return 0;
}
```

Output:

1 2 3 4 5 6 7 8 9 10

```
#include <stdio.h>
int main()
{
    char x = 0;
    for (x = 0; x <= 127; x++) {
        printf("%d ", x);
    }
    return 0;
}
```

Output:

0 1 2 3…126 127 -128 -127 ..-1 0 1 2 3 ..127..infinite loop

```
#include <stdio.h>
```

```
int main() {
    int i, a[4] = {3, 1, 2, 4}, result;
    result = a[0];
    for (i = 1; i < 4; i++) {
        if (result > a[i]) {
            continue;
        }
        result = a[i];
    }
    printf("%d", result);
    return 0;
}
```

```
void main( ) {
    int i;
for ( i = 10 ; i ; i -- )
printf ( "%d ", i ) ;
}
```

Output:

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

```
void main( ) {
    int i=5,j;
for ( scanf ( "%d", &i ) ; i <= 10 ; i++ )
printf ( "%d", i ) ;
}
```

Output:

```
5678910
```

```
#include<stdio.h>
void main( ) {
    int i,j;
for ( i=10,j=5 ; j<=6 ; j++,i-- )
   printf ( "%d %d \n", i,j ) ;
}
```

Output:

```
10 5
9 6
```

Damodar Kammili

FOR loop

```
-----------------------------------------------------------------
--

int main() {
   int i=1;
   for( printf( " Namasthe \n ") ;  i< 3 ;  printf("   CRT chaltha %d \n",i++)  )
{
          if( i==1)
                printf(" kya chaltha hai?\n " );
          else
               {
                 printf(" crt ?\n ");
                break;
               }
}
printf(" acha acha.... " );
   return 0;
}


--------------------------------------------------

-----------------------------------------------------------------
----------------------------------------------------
```

Damodar Kammili

while loop

```
#include <stdio.h>

int main() {
    int n, sum = 0, i = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &n); 10
    while (i <= n) {
        sum += i;
        i++;
    }

    printf("Sum of natural numbers up to %d is %d\n", n, sum);
    return 0;
}
```

—----------

```
#include <stdio.h>

int main() {
    int start, end;
    printf("Enter the start and end range: ");
    scanf("%d %d", &start, &end);

    printf("Even numbers between %d and %d are:\n", start, end);
    int i = start;
    while (i <= end)
  {
        if (i % 2 == 0) {
            printf("%d ", i);
          }
        i++;
    }
    printf("\n");
    return 0;
}
```

—----------

```
#include <stdio.h>

int main() {
    int num, i = 1;
    printf("Enter an integer: ");
    scanf("%d", &num);

    while (i <= 10) {
        printf("%d x %d = %d\n", num, i, num * i);
        i++;
    }
    return 0;
}
```

—----------

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[100];
    int count = 0, i = 0;
    printf("Enter a string: ");
    scanf("%s", str);

    while (i < strlen(str)) {
        char ch = str[i];
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
            count++;
        }
        i++;
```

```
    }
    printf("Number of vowels in the string is %d\n", count);
    return 0;
}
```

```
int main() {
    int n, i = 0;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    while (i < n) {
        scanf("%d", &arr[i]);
        i++;
    }
    int max = arr[0];
    i = 1;
    while (i < n) {
        if (arr[i] > max) {
            max = arr[i];
        }
        i++;
    }
    printf("The largest element is %d\n", max);
    return 0;
}
```

```
int main() {
    int num, sum = 0, digit;
    printf("Enter a number: ");
```

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

    while (num != 0) {
        sum  = sum + (num % 10)*(num % 10)*(num % 10)
        num = num /10;
    }

 while (num != 0) {
        sum  = sum + num % 10 ;
        num = num /10;
    }


    printf("Sum of the digits is %d\n", sum);
    return 0;
}
```

```
int main() {
    int num, reversed = 0, digit;
    printf("Enter a number: ");
    scanf("%d", &num);

    while (num != 0) {
        digit = num % 10;
        reversed = reversed * 10 + digit;
        num /= 10;
    }

    printf("Reversed number is %d\n", reversed);
    return 0;
}
```

```
int main() {
```

```
    int num, isPrime = 1, i = 2;
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    if (num <= 1) {
        printf("%d is not a prime number.\n", num);
        return 0;
    }

    while (i <= num / 2) {
        if (num % i == 0) {
            isPrime = 0;
            break;
        }
        i++;
    }

    if (isPrime) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}
```

—----------

```
int main() {
    int num, isPrime = 1, i = 2;
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    if (num <= 1) {
        printf("%d is not a prime number.\n", num);
        return 0;
```

```
    }

    while (i <= num / 2) {
        if (num % i == 0) {
            isPrime = 0;
            break;
        }
        i++;
    }

    if (isPrime) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}
```

```
int main() {
    int i = 0, j, rows = 4;
    while (i < rows) {
        j = 0;
        while (j < rows) {
            printf("*");
            j++;
        }
        printf("\n");
        i++;
    }
    return 0;
}
```

```
int main() {
    int i = 1, j, k, rows = 5;
    while (i <= rows) {
        j = 1;
        while (j <= rows - i) {
            printf(" ");
            j++;
        }
        k = 1;
        while (k <= 2 * i - 1) {
            printf("*");
            k++;
        }
        printf("\n");
        i++;
    }
    return 0;
}
```

—----------

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

—----------

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

—----------

BASIC level

```
#include <stdio.h>

int main() {
   int n, i = 1;
   scanf("%d", &n);
   while (i <= n)
         printf("%d ", i++);
   return 0;
}
```

—----------

```
int main() {
```

```
    int n, sum = 0, i = 1;
    scanf("%d", &n);
    while (i <= n)
        sum += i++;
    printf("%d\n", sum);
    return 0;
}
```

—----------

```
#include <stdio.h>
int main() {
    int n, i = 2;
    scanf("%d", &n);
    while (i <= n) {
        printf("%d ", i);
        i += 2;
    }
    return 0;
}
```

—----------

```
#include <stdio.h>

int main() {
    int n, i = 1;
    scanf("%d", &n);
    while (i <= n) {
        printf("%d ", i);
        i += 2;
    }
    return 0;
}
```

—----------

—----------

// Squaring of the variable

```
#include <stdio.h>

int main() {
    int n, i = 1;
    scanf("%d", &n);
    while (i <= n) {
        printf("%d^2 = %d\n", i, i * i);
        i++;
    }
    return 0;
}
```

—----------

```
#include <stdio.h>

int main() {
    int n, sum = 0, i = 2;
    scanf("%d", &n);
    while (i <= n) {
        sum += i;    // sum = sum+i ;
        i += 2;
    }
    printf("%d\n", sum);
    return 0;
}
```

—----------

```
#include <stdio.h>

int main() {
    int n, i = 5;
    scanf("%d", &n);
    while (i <= n) {
        printf("%d ", i);
        i += 5;
    }
    return 0;
}
```

—----------

```
#include <stdio.h>

int main() {
    int n, count = 0;
    scanf("%d", &n);
    while (n != 0) {
        count++;
        n /= 10;
    }
    printf("%d\n", count);
    return 0;
}
```

—----------

```
// DO-While
int main()
{
    int i = 0;
    do
```

```
    {
        printf("while vs do-while\n");
    } while (i == 0);
    printf("Out of loop");
    return 0;
}
```

Output:

While vs do-while *infinite times*

```
int main( )
{
int i=0, j=0;
while( i<4,j<5)
{
i++;
j++;
}
printf("%d,%d\n", i, j );
return 0;
}
```

```
#include <stdio.h>
int main()
{
    int c = 1;
    while (c <= 5)
    {
        if (c == 3)
            break;
        printf("%d ", c);
        c++;
    }
    return 0;
}
```

Output:

1 2

—----------

—----------

// **What will happen when you try to execute this program in TurboC7 and 32 bit os systems**

```
void main( )
{ int i=1;
 while (i<=32768){
 printf ("%d  ",i );
 i = i + 1 ;
 }
}
```

Solution :

In TurboC7 : The program went into infinite loop

In 32 bit OS systems : The program will print numbers from 1 to 32767

—----------

What will be the output of the following program?

```
 void main( ) {
 int i = 1 ;
 while ( i<=10 ) ;
   {
   printf (" %d \n ", i );
    i = i + 1 ;
    }
```

Output:

Infinite loop

—----------

```
 void main( ) {
 int i = 1 ;
 while ( i++ < 10 )
  {
   printf (" %d \n ", i );
    }
```

Output:

2 3 4 5 6 7 8 9 10

—----------

```
void main( )
{  int sem = 1;
while(sem<=8)
{
  if( sem==1){
printf( " Hi! Learn me in this semester " );
break;
}
else
printf(" I will not let you understand Java " );
sem++;
}
```

Output:

Hi! Learn me in this semester

—----------

```
void main( ) {
int i = 1 ;
while (i<=10)
      i++;
     printf("%d ", i);
}
```

Output:

11

—----------

```
void main( ) {
int i = 1 ;
while (i>=5)
      i++;
     printf("%d ", i);
}
```

Output:

1

Damodar Kammili

—----------

—----------

—----------

—----------

—----------

—----------

Switch programs

```
int main()
{
   switch (printf("DD"))                    // hint : Answer lies in the printf statement
   {
   default:
      printf("Guwahati");
   case 1:
      printf("Delhi");
      break;
   case 2:
      printf("Kharagpur");
      break;
   case 3:
      printf("Madras");
      break;
   }
   return 0;
}
```

Output:

Kharagpur

—----------

// Finding whether an alphabet is **a vowel or consonant.**

```
int main() {
printf(" Enter any alphabet ");
scanf("%c", &alpha);
   switch (alpha) {
      case 'a':
      case 'e':
      case 'i':
      case 'o':
      case 'u': printf(" Vowel");
```

```
                    break;
     default: printf(“Consonant”);
   }
   return 0;
}
```

—----------

```
#include<stdio.h>
int main() {
   int a[] = {1,2,3,4};
   switch (sizeof(a)) {
      case 4:  break;
      case 8 : break;
      case 16: printf("DSA ");
               break;
    }
   printf("Group");
   return 0;
}
```

Output:

DSA Group

—----------

```
int main() {
   int i = 0;
   char c = 'a';
   while (i < 5) {
      i++;
      switch (c) {
         case 'a':
            printf("%c ", c);
            break;
      }
   }
   printf("a\n");
```

```
    return 0;
}
```

output:

a a a a a a

—----------

```
void main( )
{
 int ch = 'a' + 'b' ;
 switch ( ch )
 {
 case 'a' :
 case 'b' :
 printf ( "\nYou entered b" ) ;
 case 'b' + 'a' :
 printf ( "\nYou entered a and b" ) ;
 }
}
```

Output:

You entered a and b

—----------

```
void main( )
{
 int i=0 , j;
 switch ( j= i ) ;
 {
   case 0: printf ( "\nClub" ) ;
 }
return 0;
}
```

Output:

Compilation ERROR - case 0 outside of switch.

—----------

—----------

—----------

Damodar Kammili

—----------

Arrays

How Array elements are being stored consecutively in the memory.

```
void main( )
{
int num[ ] = { 10, 20, 30, 40, 50 } ;
int i ;
for ( i = 0 ; i <= 5 ; i++ )
 {
 printf ( "\nelement  %d ", i ) ;
 printf ( "at address = %u", &num[i] ) ;
 }
}
```

Output: (executed in a 64 bit system)

```
element  0 at address = 2242260416
element  1 at address = 2242260420
element  2 at address = 2242260424
element  3 at address = 2242260428
element  4 at address = 2242260432
element  5 at address = 2242260436
```

—---

// example for arrays containing integers **while assigning Characters.**

```
void main( )
{
 int array[26], i ;
 for ( i = 0 ; i <= 5 ; i++ )
 {
 array[i] = 'A' + i ;
 printf ( "\n%d %c  ", array[i], array[i] ) ;
 }
}
```

Output:

65 A

66 B

67 C

68 D

69 E

70 F

—----------

// **Example for Array out of bounds**

```
void main( )
{
int num[40], i ;
for ( i = 0 ; i <= 100 ; i++ )
num[i] = i ;
}
```

Output:

Segmentation fault

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

// **What could be the final values of i, j, k after execution?**

```
 int main() {
    int arr[] = {1, 2, 3, 4, 5, 6 };
    int i, j, k;
    j = 0;
    j = ++arr[2];
    k = arr[1]++;
    i = arr[j++];
    printf("i=%d, j=%d, k=%d", i, j, k);
    return 0;
}
```

-

Output:

i=5 j=5 k=2

—----------

```
int main() {
    int a;
    int arr[5] = {1, 2, 3, 4, 5};
    arr[1] = ++arr[1];
    a = arr[1]++;
    arr[1] = arr[a++];
    printf("%d,%d", a+1, arr[4]);
    return 0;
}
```

Output:

5,5

—----------

—----------

—----------

—----------

Two - Dimensional Arrays

```
//Below program demonstrates  how the first dimension acts ?
void main( )
{
int num[2 ][ 2 ] = { 10, 20, 30, 40 } ;
int i ;
printf("%u  ",num[i]);
printf("%u  ",num[i+1]);
}
```

Output:

3613663888 3613663896 // these values may differ for you

—----------

—----------

—----------

—----------

Pointers and Arrays:

```
void main( )
{
 int b[ ] = { 10, 20, 30, 40, 50 } ;
 int i ;
 for ( i = 0 ; i < 4 ; i++ )
 printf ( " %d", *(b+i) ) ;
}
```

Output:

10 20 30 40

—----------

```
void main( )
{
 int b[ ] = {11 0, 200, 30, 40, 500 } ;
 int i, *k ;
 k = b ;
```

```
 for ( i = 0 ; i <= 4 ; i++ )
{
   printf ( "  %d" *k ) ;
   k++ ;
}
}
```

Output:

```
 110 20 30 40 500
```

—----------

```
void main( )
{
   int *arr[4] ;
int i = 35, j = 5, k = 20, l = 72, m ;
arr[0] = &i ;
arr[1] = &j ;
arr[2] = &k ;
arr[3] = &l ;
for ( m = 0 ; m <= 3 ; m++ )
printf ( "%d ", * ( arr[m] ) ) ;
}
```

Output:

```
31 5 19 71
```

—----------

```
void main( )
{
  static int a[ ] = { 0, 1, 2, 3, 4 } ;
  int *p[ ] = { a, a + 1, a + 2, a + 3, a + 4 } ;
  printf ( "\n%u %u %d", p, *p, * ( *p ) ) ;
}
```

Output:

```
545988192  4210720  0    // first two values are different for you
```

—----------

—----------

—----------

Multi-Dimensional Arrays with pointers:

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

Strings

—----------

---—--

// Printing the string

```
void main( )
{
char seminar[ ] = "birds_robbery" ;
int i = 0 ;
    while ( i < 13 )
    {
        printf ( "%c", seminar[i] ) ;
        i++ ;
   }
}
```

Output:

birds_robbery

—----------

// Printing the string using %s format specifier

```
//The %s used in printf( ) is a format specification for printing out a string
void main( )
{
 char name[ ] = "Code_slashing" ;
 printf ( "%s", name ) ;
}
```

Output:

Code_slashing

—----------

```
int main() {
   char str1[] = "Quiz-0-Assignment";
   char str2[] = {'Q, 'u', 'i', 'z', '-', '0', '-', 'A', 's', 's', 'i', 'g', 'n', 'm', 'e', 'n', 't'};
   int n1 = sizeof(str1) / sizeof(str1[0]);
   int n2 = sizeof(str2) / sizeof(str2[0]);
   printf("n1 = %d, n2 = %d", n1, n2);
   return 0;
}
```

Output:

n1= 18 n2=17

—----------

// Concatenating and Copying one string into another string

```
#include <stdio.h>
#include <string.h>

int main() {
   char str1[20] = "Hello", str2[20] = " world";
   printf("%s", strcpy(str2, strcat(str1, str2)));
   return 0;
}
```

Output:

Hello world

—----------

// Fill the blank to get the output as " Hwi orea "

```
int main() {
   int i;
   char s[] = "How is your exam";
   for (i = 0; s[i] != '\0'; ++i) {
      if ( —----------------------)
     {
         printf("%c", s[i]);
      }
   }
   return 0;
}
```

solution :

i%2==0

—----------

```
#include <stdio.h>
#include <string.h>
```

```
int main() {
   static char str1[] = "_dd";
   static char str2[20];
   static char str3[] = "MIC";
   int i;
   i = strcmp(strcat(str3, strcpy(str2, str1)), "MIC_dd");
   printf("i = %d\n", i);
   return 0;
}
```

Output:

0

—----------

//What could be the output of the following program?

```
#include <stdio.h>
#include <string.h>

int main()
{
   char p[] = "assignment";
   char t;
   int i, j;
   for (i = 0, j = strlen(p); i < j; i++)
   {
      t = p[i];
      p[i] = p[j - i];
      p[j - i] = t;
   }
   printf("%s", p);
   return 0;
}
```

Output:

Nothing will be printed

—----------

// **Characters and functions**

```
void fu(int, int);
int main()
{
   char x = 67, y = 'C';
   fu(x, y);
   return 0;
}

void fu(int x, int y)
{
   printf("%d,%d", x, y);
}
```

Output:

67 67

—----------

// **Finding the length of the string without using the built in function**

```
int main() {
   char str[] = "Hello, world!";
   int length = 0;

   while (str[length] != '\0') {
      length++;
   }
   printf("Length of the string = %d\n", length);
   return 0;
}
```

Output:

Length of the string = 13

—----------

```
void main( )
{
 char str1[ ] = { 'H', 'e', 'l', 'l', 'o' } ;
 char str2[ ] = "Hello" ;

 printf ( "\n%s", str1 ) ;
 printf ( "\n%s", str2 ) ;
}
```

Output:

Hello.

Hello

—----------

—----------

—----------

—----------

Damodar Kammili

—----------

—----------

—----------

—----------

—----------

—----------

—----------

functions

Damodar Kammili

Functions

```
#include <stdio.h>

// Function to add two numbers
int add(int a, int b) {
    return a + b;
}

int main() {
    int num1, num2, sum;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    sum = add(num1, num2); // Calling the function
    printf("The sum is: %d\n", sum);
    return 0;
}
```

```
#include <stdio.h>

// Function to calculate factorial
int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Factorial of %d is %d\n", num, factorial(num)); //
Calling the function
    return 0;
}
```

```
#include <stdio.h>

// Function to check even or odd
void checkEvenOdd(int num) {
    if (num % 2 == 0) {
        printf("%d is Even\n", num);
    } else {
```

```
        printf("%d is Odd\n", num);
    }
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    checkEvenOdd(num); // Calling the function
    return 0;
}
```

```
#include <stdio.h>

// Function to swap two numbers
void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main() {
    int a, b;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    printf("Before swapping: a = %d, b = %d\n", a, b);
    swap(&a, &b); // Calling the function
    printf("After swapping: a = %d, b = %d\n", a, b);
    return 0;
}
```

```
#include <stdio.h>
#define PI 3.14159

// Function to calculate area of a circle
float areaOfCircle(float radius) {
    return PI * radius * radius;
}

int main() {
    float radius;
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);
```

```
    printf("Area of the circle: %.2f\n", areaOfCircle(radius)); //
Calling the function
    return 0;
}
```

SIMPLE

```
#include <stdio.h>

// Function to print "Hello, World!"
void printMessage() {
    printf("Hello, World!\n");
}

int main() {
    printMessage(); // Calling the function
    return 0;
}
```

```
#include <stdio.h>

// Function to add two numbers
int add() {
    return 2 + 3;
}

int main() {
    int result = add(); // Calling the function
    printf("The sum is: %d\n", result);
    return 0;
}
```

```
#include <stdio.h>

// Function to print a number
void printNumber(int n) {
    printf("The number is: %d\n", n);
}

int main() {
    printNumber(5); // Calling the function
```

```
    return 0;
}
```

--

--
```
#include <stdio.h>

// Function definition
int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}

int main() {
    int num1, num2;

        printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    // Function call to find maximum
    printf("The maximum number is: %d\n", max(num1, num2));
    return 0;
}
```

--
```
#include <stdio.h>

// Function definition
int square(int num) {
    return num * num;
}

int main() {
    int num, result;

    // Input from the user
    printf("Enter a number: ");
    scanf("%d", &num);

    // Function call to calculate square
    result = square(num);
```

```
    // Output the result
    printf("The square of %d is: %d\n", num, result);
    return 0;
}
```

```
--------------------------------------------------------------------
--
#include <stdio.h>

// Function definition
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int num1, num2;

    // Input from the user
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    // Function call to swap numbers
    swap(&num1, &num2);

    // Output the result
    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);
    return 0;
}
```

```
--------------------------------------------------------------------
--
```

```
--------------------------------------------------------------------
--
--------------------------------------------------------------------
--
--------------------------------------------------------------------
--
#include <stdio.h>

// Function to print "Goodbye!"
void sayGoodbye() {
    printf("Goodbye!\n");
}
```

```
int main() {
    sayGoodbye(); // Calling the function
    return 0;
}
```

```
// Find the error in the following program ?
#include <stdio.h>

int f(int a)
{
    a > 20? return(10): return(20);
}

int main()
{
    int b;
    b = f(20);
    printf("%d\n", b);
    return 0;
}
```

solution:

'Return' statement cannot be used with the conditional operators

---

```
// How many times 'Hi' will be printed?
#include <stdio.h>
int i;
int fun();

int main()
{
    while(i)
    {
        fun();
        main();
    }
    printf("Hello\n");
    return 0;
}

int fun()
{
```

```
    printf("Hi");
}

output :
```

Zero times

```
--------------------------------------------------------------------
--

#include <stdio.h>
float func(float age[]);

int main()
{
    float result, age[] = {23.4, 55, 22.6, 3, 40.5, 18};
    result = func(age);
    printf("%.2f", result);
    return 0;
}

float func(float age[])
{
    int i;
    float result, sum = 0.0;
    for (i = 0; i < 6; ++i)
    {
        sum += age[i];
    }
    result = (sum / 6);
    return result;
}
```

Output:

27.08

```
--------------------------------------------------------------------
--
// what will be printed in the output screen?
#include <stdio.h>
int main()
{
    int i;
    for (i = 0; i < 5; i++)
    {
        int i = 10;
        printf("%d ", i);
        i++;
    }
```

```
    return 0;
}
```

Output:

```
10 10 10 10 10
```

```
// What this function is finding ?

int func(int a, int b, int c)
{
    if ((a >= b) && (c < b)) return b;
    else if (a >= b) return func(a,c,b);
    else return func(b,a,c);
}
```

Output:

```
It is finding the middle number of a,b,c
```

```
// what will happen when we provide j as 50?

int f(int j)
{
    static int i = 50;
    int k;
    if (i == j)
    {
        printf("something");
        k = f(i);
        return 0;
    }
    else return 0;
}
```

solution:

The function will exhaust the runtime stack or run into an infinite loop when j = 50

```
#include <stdio.h>
```

```
int fun(int arr[]) {
    arr = arr + 2;
    printf("%d ", arr[0]);
}

int main() {
    int arr[3] = {7, 11, 19};
    fun(arr);
    printf("%d ", arr[0]);
    printf("%d ", arr[1]);
    return 0;
}
```

Output :

```
19 7 11
```

Case scenario for how Functions are used.

There are basically two types of functions:

Library functions Ex. printf( ), scanf( ) etc.

User-defined functions Ex.

fibonacci( ), sum( ) etc.

- Any function can be called from any other function
- A function can be called any number of times
- A function can call itself. Such a process is called 'recursion'.

```
#include <stdio.h>
```

```c
// Function declarations
void greetUser();
int add(int a, int b);
int multiply(int a, int b);
int findMax(int a, int b);

int main() {
    int num1 = 10, num2 = 5;
    int sum, product, bigger;

    greetUser();  // Just a greeting

    // Call functions that return values
    sum     = add(num1, num2);
    product = multiply(num1, num2);
    bigger  = findMax(num1, num2);

    // Display results
    printf(" You gave me: %d and %d\n", num1, num2);
    printf(" Sum: %d\n", sum);
    printf(" Product: %d\n", product);
    printf(" Bigger number: %d\n", bigger);

    return 0;
}

// Function definitions

void greetUser() {
    printf(" Hello! I'm your friendly calculator.\n");
    printf("Let's do some simple math with two numbers.\n\n");
}

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

int multiply(int a, int b) {
    return a * b;
}

int findMax(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
```

```
}
```

The core idea is to show that functions are like mini-programs that do specific tasks, making your main program cleaner and easier to manage.

// shall we include this into book, so that they can understand internally how functions are used in the software development.

Case 2 : USer Authentication.

```
#include <stdio.h>
#include <string.h>

void displayWelcomeMessage();
int authenticateUser();
void performUserActions();
void displayLoginFailure();

int main() {
    displayWelcomeMessage();

    int login_success = authenticateUser();

    if (login_success) {
        printf("\nLogin successful! Welcome.\n");
        performUserActions();
    } else {
        displayLoginFailure();
    }

    printf("\nProgram Exiting. Goodbye!\n");
```

```
    return 0;
}




void displayWelcomeMessage() {
    printf("=======================================\n");
    printf("  Welcome to the Secure System Portal!  \n");
    printf("=======================================\n");
}

int authenticateUser() {
    char username[20];
    char password[20];
    const char correct_username[] = "admin";
    const char correct_password[] = "securepass";
    int attempts = 0;
    const int max_attempts = 3;

    while (attempts < max_attempts) {
        printf("\n--- Login Required ---\n");
        printf("Enter username: ");
        scanf("%19s", username);
        printf("Enter password: ");
        scanf("%19s", password);
 if (strcmp(username, correct_username) == 0&& strcmp(password, correct_password) == 0) {
            return 1;
        } else {
            printf("Invalid credentials. Please try again.\n");
            attempts++;
            printf("Attempts left: %d\n", max_attempts - attempts);
        }
    }
    return 0;
}
```

```
void performUserActions() {
    printf("\n--- User Dashboard ---\n");
    printf("1. View Profile\n");
    printf("2. Change Settings\n");
    printf("3. Access Secret Data\n");
    printf("----------------------\n");
    printf("Performing important actions...\n");
    printf("Data loaded successfully.\n");
}


void displayLoginFailure()
 {
    printf("\nMaximum login attempts exceeded. Access denied.\n");
    printf("Please contact support if you believe this is an error.\n");
}
```

Recursion,storage classes

What is the value of the result variable, when it is called from main?

result = **func**(**5**);

```
int func( int  n )
{  int k, sum=0 ;
   if( n < 0 )
     return  1 ;
for ( k=0; k < n; k++)
  sum += func(n-1)+func(n-2);
return sum;
}
```

Output :

265

—----------

```
// Recursive function definition
unsigned int func(unsigned int n, unsigned int r) {
   if (n > 0) {
      return (n % r + func(n / r, r));
   } else {
      return 0;
   }
}

int main() {
     printf("The return value of func(513, 2) is: %u \n", func( 513, 2 ) );
   return 0;
}
```

Output:

2

—----------

—----------

// **How many recursive calls can be made when we call**

```
#include<stdio.h>
int get(int n) {
   if (n < 0)
      return 1; // Base Case 1
   if (n < 2)
      return get(n - 1); // Recursive call 1
   else
      return get(n - 2); // Recursive call 2
}

void main( )
{ int get (int );
  int result = 0;
result = get(5);
printf("result = %d", result);
}
```

Output:

result = 1

The number of recursive calls = 5, from main function to till the base case hit.

—----------

// **what is the output of the following program?**

```
#include <stdio.h>
int f(int);
void main( )
{ int result=0;
result = f(1);
printf("%d ",result);
}
int f(int n)
{
   static int i = 1;
   if (n >= 5)
```

```
        return n;
    n = n + i;
    i++;
    return n+f(n);
}
```

Output:

20

—----------

—----------

Main Purpose of Recursion

Recursion is used to **solve problems that can be broken down into smaller subproblems** of the **same** type. It allows a function to call itself to solve those subproblems elegantly.

Base Case in Recursion :

The **base case** is the condition in a recursive function that **stops the recursion**.

—----------

// Factorial of a number

```
int factorial(int n) {
    if (n == 0) return 1;  // Base case
    return n * factorial(n - 1);
}

int main() {
    int n = 5;
    printf("Factorial of %d = %d\n", n, factorial(n));
    return 0;
}
```

Output:

Factorial of 5 = 120

—----------

// Fibonacci series

// yet to write..

Output:

—----------

// sum of Digits

```
int sumOfDigits(int n) {
    if (n == 0) return 0;
    return (n % 10) + sumOfDigits(n / 10);
}

int main() {
    int num = 1234;
    printf("Sum of digits of %d = %d\n", num, sumOfDigits(num) );
    return 0;
}
```

Output:

Sum of digits of 1234 = 10

—----------

// Reversing a number

```
int reverse(int n, int rev) {
    if (n == 0) return rev;
    return reverse(n / 10, rev * 10 + (n % 10) );
}

int main() {
    int num = 1234;
    printf("Reversed number: %d\n", reverse(num, 0) );
    return 0;
```

```
}
```

Output:

```
Reversed number: 4321
```

// GCD (Greatest Common Divisor)

```
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

int main() {
    int a = 48, b = 18;
    printf("GCD of %d and %d = %d\n", a, b, gcd(a, b) );
    return 0;
}
```

Output:

```
GCD of 48 and 18 = 6
```

// Power Function (a^b)

```
int power(int a, int b) {
    if (b == 0) return 1;
    return a * power(a, b - 1);
}

int main() {
    int a = 2, b = 5;
    printf("%d^%d = %d\n", a, b, power(a, b) );
    return 0;
}
```

Output:

```
2^5 = 32
```

// Check if a string is a palindrome or not ?

```
#include <string.h>

int isPalindrome(char str[], int start, int end) {
    if (start >= end) return 1;
    if (str[start] != str[end]) return 0;
    return isPalindrome(str, start + 1, end - 1);
}

int main() {
    char str[] = "madam";
    if (isPalindrome(str, 0, strlen(str) - 1))
        printf("%s is a palindrome\n", str);
    else
        printf("%s is not a palindrome\n", str);
    return 0;
}
```

Output:

madam is a palindrome

—----------

—----------

// Printing Numbers from N to 1 (Descending order)

```
void printDescending(int n) {
    if (n == 0) return;
    printf("%d ", n);
    printDescending(n - 1);
}

int main() {
    int n = 5;
    printf("Numbers from %d to 1:  ", n);
    printDescending(n);
    return 0;
}
```

Output:

Numbers from 5 to 1 : 5 4 3 2 1

—----------

// Printing Array elements recursively

```
void printArray(int arr[], int index, int size) {
    if (index >= size) return;
    printf("%d ", arr[index]);
    printArray(arr, index + 1, size);
}

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Array elements: ");
    printArray(arr, 0, size);
    return 0;
}
```

Output:

Array elements: 10 20 30 40 50

—----------

// Checking whether a number is prime or not (recursive division test)

```
int isPrime(int n, int i) {
    if (n <= 2)
        return (n == 2);
    if (n % i == 0)
        return 0;
    if (i * i > n)
        return 1;
    return isPrime(n, i + 1);
}

int main() {
    int num = 29;
    if (isPrime(num, 2))
```

```
        printf("%d is a prime number\n", num);
    else
        printf("%d is not a prime number\n", num);
    return 0;
}
```

Output:

29 is a prime number

// Count Digits in a number

```
int countDigits(int n) {
    if (n == 0) return 0;
    return 1 + countDigits(n / 10);
}

int main() {
    int num = 12345;
    printf("Number of digits in %d = %d\n", num, countDigits(num));
    return 0;
}
```

// Finding Maximum number in an array recursively.

```
int findMax(int arr[], int n) {
    if (n == 1) return arr[0];
    int max_rest = findMax(arr, n - 1);
    return (arr[n - 1] > max_rest) ? arr[n - 1] : max_rest;
}

int main( ) {
    int arr[] = {12, 45, 23, 67, 34};
```

```
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Maximum element = %d\n", findMax(arr, size) );
    return 0;
}
```

Output:

Maximum element = 67

Visualization (call stack) :

findMax(arr, 5)

→ findMax(arr, 4)

→ findMax(arr, 3)

→ findMax(arr, 2)

→ findMax(arr, 1) // returns 12

↑ compare 45,12 → 45

↑ compare 23,45 → 45

↑ compare 67,45 → 67

↑ compare 34,67 → 67

—----------

—----------

—----------

STORAGE CLASSES

—----------

There are four storage classes in C:

(a) Automatic storage class

(b) Register storage class

(c) Static storage class

(d) External storage class

—----------

Automatic Storage Class

The features of a variable defined to have an automatic storage class are as under:

Storage – Memory.

Default initial value – An unpredictable value, which is often called a garbage value.

Scope – Local to the block in which the variable is defined.

Life – Till the control remains within the block in which the variable is d

Example

```
void main( )
{
 auto int i, j ;
 printf ( "\n%d %d", i, j ) ;
}
```

Output:

232343 332 // these are garbage values.

```
void main( )
{
 auto int i = 1 ;
 {
    {
          {
           printf ( "\n%d ", i ) ;
          }
       printf ( "%d ", i ) ;
     }
 printf ( "%d", i ) ;
 }
}
```

OUTPUT:

1 1 1

—----------

```
void main( )
{
```

```
auto int i = 1 ;
{
   auto int i = 2 ;
   {
     auto int i = 3 ;
     printf ( "\n%d ", i ) ;
   }
   printf ( "%d ", i ) ;
}
printf ( "%d", i ) ;
}
```

Output:

3 2 1

Register Storage Class

The features of a variable defined to be of register storage class are as under:

Storage - CPU registers.

Default initial value - Garbage value.

Scope - Local to the block in which the variable is defined.

Life - Till the control remains within the block in which the variable is defined.

A value stored in a CPU register can always be accessed faster than the one that is stored in memory. Therefore, if a variable is used at many places in a program it is better to declare its storage class as register. A good example of frequently used variables is loop counters. We can name their storage class as register

```
#include<stdio.h>


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

Output:

1 2 3 4 5 6 7 8 9 10

—----------

Static Storage Class

The features of a variable defined to have a static storage class are as under:

Storage – Memory.

Default initial value – Zero.

Scope – Local to the block in which the variable is defined.

Life – Value of the variable persists between different function calls.

—----------

External Storage Class

The features of a variable whose storage class has been defined as external are as follows:

Storage – Memory.

Default initial value – Zero.

Scope – Global.

Life – As long as the program's execution doesn't come to an end.

Example -1

```
int x = 10 ;
 main( ) {
       extern int y ;
      printf ( "\n%d %d", x, y ) ;
}
int y = 20 ;
```

Output:

10 20

—----------

Example-2

—----------

```
int i;
```

```
void increment();
void decrement();
int main()
{
   printf("\nInitial value of i = %d", i);

   increment();
   increment();
   decrement();
   decrement();
   return 0;
}
void increment()
{
   i = i + 1;
   printf("\nOn incrementing i = %d", i);
}

void decrement()
{
   i = i - 1;
   printf("\nOn decrementing i = %d", i);
}
```

Output:

```
Initial value of i = 0
On incrementing i = 1
On incrementing i = 2
On decrementing i = 1
On decrementing i = 0
```

—----------

```
#include <stdio.h>
void func();
void  main( )
```

```
{
 func( ) ;
 func( ) ;
}

void func( )
{
 auto int i = 0 ;
 register int j = 0 ;
 static int k = 0 ;
 i++ ; j++ ; k++ ;
 printf ( "\n %d % d %d", i, j, k ) ;
}
```

Output:

1 1 1

1 1 2

—----- ----------

Sir , please look into this problem manually and executed part ,

When I want to try this **static variable in the recursion**, the expected output is not as expected with the executed one..

```
 void main( )
{
    int result;
   int func();
  result = func( );
printf("\nresult =%d " ,result);
}

int func( )
{
static int k = 1;
printf("%d ",k);
if ( k > 5 )
     return 0;
```

```
k++;
return k+func();
}
```

Output:

```
1 2 3 4 5 6
Result = 30
```

—---

—---

—---

—---

—---

Programs on Patterns

Damodar Kammili

Programs on Patterns

// print Right angled triangle

```
#include <stdio.h>

int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

// printing a Square

```
#include <stdio.h>

int main() {
    int rows = 4;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < rows; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

Basic level

```
#include <stdio.h>

int main() {
    int n=5;

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

    return 0;
}
```

Output:

```
1 2 3 4 5
```

```
#include <stdio.h>

int main() {
    int n=5, sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
   printf("Sum of first %d natural numbers is: %d\n", n, sum);

    return 0;
}
```

Output:

```
Sum of first 5 natural numbers is: 15
```

// printing Even Numbers

```
#include <stdio.h>
int main() {
    int n=10;

    printf("Even numbers from 1 to %d are: ", n);
```

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

    return 0;
}
```

Output:

```
Even numbers from 1 to 10 are : 2 4 6 8 10
```

// printing Odd numbers

```
#include <stdio.h>
int main() {
    int n = 10 ;
printf("Odd numbers from 1 to %d are: ", n);
    for (int i = 1; i <= n; i += 2) {
        printf("%d ", i);
    }
return 0;
}
```

Output:

```
Odd numbers from 1 to 10 are: 1 3 5 7 9
```

// Square of a number

```
#include <stdio.h>

int main() {
    int n = 5 ;
    printf("Squares of numbers from 1 to %d are:\n", n);
    for (int i = 1; i <= n; i++) {
        printf("%d^2 = %d\n", i, i * i);
    }

    return 0;
}
```

Output:

```
Squares of numbers from 1 to 5 are:
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
```

// print sum of even numbers

```
#include <stdio.h>

int main() {
    int n, sum = 0;

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

    for (int i = 2; i <= n; i += 2) {
        sum += i;
    }

    printf("Sum of even numbers up to %d is: %d\n", n, sum);

    return 0;
}
```

// Numbers divisible by 3

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Numbers divisible by 3 up to %d are:\n", n);



for (int i = 1; i <= n; i++) {
        if (i % 3 == 0) {
           printf("%d ", i);
        }
    }
    return 0;
}
```

// 5 Multiples

```
#include <stdio.h>

int main() {
    int n;

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

    printf("Multiples of 5 up to %d are:\n", n);
    for (int i = 5; i <= n; i += 5) {
        printf("%d ", i);
    }

    return 0;
}
```

// Counting digits in a number

```
#include <stdio.h>
int main() {
    int n, count = 0;
    printf("Enter a number: ");
    scanf("%d", &n);
    for (int temp = n; temp != 0; temp /= 10) {
        count++;
    }
    printf("The number of digits in %d is: %d\n", n, count);

    return 0;
}
```

```
******  →6
* * * * * * →6
```

```
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
```

Damodar Kammili

```
*
*
*
*
```

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

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

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

```
0 1 1 1 1
2 0 2 2 2
3 3 0 3 3
4 4 4 0 4
5 5 5 5 0
```

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

```
1
2 3
```

```
4 5 6
7 8 9 10

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

// Print Pyramid

```
#include <stdio.h>
int main() {
    int rows = 5;

 for (int i = 1; i <= rows; i++)
{
        for (int j = 1; j <= rows - i; j++)
        {
            printf(" ");
        }
        for (int k = 1; k <= (2 * i - 1); k++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

// Print Numbers in a triangle

```
#include <stdio.h>

int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
```

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

Output:

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

Damodar Kammili

Pointers

```
main( ) {
        int i = 3 ;
        printf ( "\nAddress of i = %u", &i ) ; // for decimal address
        printf ( "\nValue of i = %d", i ) ;
}
```

Output:

Address of i =924491740 // this value varies depending on the system.

Value of i = 3

—----------

```
void main( )
{
int i = 3  , *j ;
p = &i ;
printf ( "\nValue of i = %d", i ) ;
printf ( "\nAddress of i = %u", &i ) ;
printf ( "\nAddress of i = %p", j ) ;
printf ( "\nValue of i = %d", *( &i ) ) ;
}
```

Output:

Address of i = 4190806532

Address of i = 0x7ffff9caa204

Value of i = 3

Value of i = 3

// As you can see, i's value is 3 and p 's value is i's address.

—----------

```
void main( )
{
 int i = 3 , *j ;
 j = &i ;
 printf ( "\nAddress of i = %u", &i ) ;
 printf ( "\nAddress of i = %u", j ) ;
 printf ( "\nAddress of j = %u", &j ) ;
 printf ( "\nValue of j = %u", j ) ;
```

```
 printf ( "\nValue of i = %d", i ) ;
 printf ( "\nValue of i = %d", *( &i ) ) ;
 printf ( "\nValue of i = %d", *j ) ;
}
```

Output:

```
Address of i = 2951971756
Address of i = 2951971756
Address of j = 2951971744
Value of j = 2951971756
Value of i = 3
Value of i = 3
Value of i = 3
```

—----------

```
#include<stdio.h>

void main( )
{
 int i = 3, *j, **k ;

 j = &i ;
 k = &j ;
 printf ( "\nAddress of i = %u", &i ) ;
 printf ( "\nAddress of i = %u", j ) ;
 printf ( "\nAddress of i = %u", *k ) ;
 printf ( "\nAddress of j = %u", &j ) ;
 printf ( "\nAddress of j = %u", k ) ;
 printf ( "\nAddress of k = %u", &k ) ;
 printf ( "\nValue of j = %u", j ) ;
 printf ( "\nValue of k = %u", k ) ;
 printf ( "\nValue of i = %d", i ) ;
 printf ( "\nValue of i = %d", * ( &i ) ) ;
 printf ( "\nValue of i = %d", *j ) ;
 printf ( "\nValue of i = %d", **k ) ;
```

```
}
```

Output:

```
Address of i = 3741546924
Address of i = 3741546924
Address of i = 3741546924
Address of j = 3741546912
Address of j = 3741546912
Address of k = 3741546904
Value of j = 3741546924
Value of k = 3741546912
Value of i = 3
Value of i = 3
Value of i = 3
Value of i = 3
```

—----------

```
#include<stdio.h>

void main( )
{

 float a = 13.5 ;
 float *b, *c ;
 b = &a ; /* suppose address of a is 1006 */
 c = b ;
 printf ( "\n%u %u %u", &a, b, c ) ;
 printf ( "\n%f %f %f %f %f", a, *(&a), *&a, *b, *c ) ;
}
```

Output:

```
3397749612  3397749612  3397749612
13.500000  13.500000  13.500000  13.500000  13.500000
```

—----------

call by reference (example)

—----------

—----------

—----------

—----------

```
#include <stdio.h>

int main()
{
   char *s = "programming";
   char *p = s;
   printf("%c,%c", *(p + 3), s[3] );
   return 0;
}
```

Output :

g g

—----------

```
#include <stdio.h>

int main()
{
   short num[3][2] = {{2,5},{11,17},{23,28}}; // Correct initialization for a 2D array
   printf("%d,%d", *(num+2)[0], **(num+1));
   return 0;
}
```

Output :

23 11

—----------

```
#include <stdio.h>

int main()
{
   int ary[4] = {1, 2, 3, 4};
   int *p;
   p = ary + 3; // 'p' now points to the element at index 3 of 'ary'
   *p = 5;      // The value at the memory location pointed to by 'p' (which is ary[3]) is changed
to 5
   printf("%d\n", ary[3]);
   return 0;
}
```

Output:

5

—----------

```
#include <stdio.h>

int main()
{
   int *ptr, a = 7;
   ptr = &a;    // ptr now holds the memory address of 'a'
   *ptr = *ptr - 2; // The value at the memory address pointed to by ptr (which is 'a') is updated.
                    // It takes the current value of 'a' (7), subtracts 2, so a becomes —?
   printf("%d,%d ", *ptr, a);
   return 0;
}
```

Output :

5 5

—----------

```
#include <stdlib.h>

int main()
{
   int i;
   int *ptr = (int *) malloc(5 * sizeof(int));

   for (i = 0; i < 5; i++)
   {
      *(ptr + i) = i;
   }

   printf("%d ", *ptr++);
   printf("%d ", (*ptr)++ );
   printf("%d ", *ptr);
   printf("%d ", *++ptr);
   printf("%d ", ++*ptr);

   // It's good practice to free dynamically allocated memory
   free(ptr - 4); // Adjust ptr back to the original start before freeing

   return 0;
}
```

Output :

```
0 1 2 2 3
```

—----------/

/ What is the output of the following C code? Assume that the address of x is 2000(in decimal) and an integer requires four bytes of memory?

```
#include <stdio.h>

int main() {
   unsigned int x[4][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};
   printf("%u, %u, %u", x + 3, *(x + 3), *(x + 2) + 3);
   return 0;
}
```

Output:

2036 2036 2036

—----------

// In which condition " Hello world " will be printed?

```
#include <stdlib.h>

int main() {
   int *ptr;
   ptr = (int *)malloc(sizeof(int) * 10);
   if (ptr == NULL) {
      printf("Hello world!\n");
   }
   return 0;
}
```

Output:

if the memory could not be allocated to the pointer "ptr"

—----------

// The program allocates —----------bytes to ptr. Assume sizeof(int) =4

```
#include <stdio.h>
#include <stdlib.h>

int main() {
   int *ptr;
   ptr = (int*)malloc(sizeof(int) * 4);
   ptr = realloc(ptr, sizeof(int) * 2);
```

```
    return 0;
}
```

Output:

8

—----------

```
// What does fp point to in the program?

int main() {
    FILE *fp;
    fp = fopen("hello", "r");
    return 0;
}
```

Output:

> A structure which contains a char pointer which points to the first character of a file.

—----------

```
// What does the output of the following program ?

#include <stdio.h>

int main() {
    int *p, a = 100;
    p = &100; // This line has an error: cannot take address of a literal
    printf("%d", *p);
    return 0;
}
```

Output:

Compilation error

—----------

// What is the output of the following C program ?

```
#include <stdio.h>
struct p {
   int x;
   char y;
};
int main() {
   struct p p1[] = {{1, 21}, {69, 42}, {64, 0}};    // Corrected initialization based on structure
   struct p *ptr1 = p1;
    int x = (sizeof(p1) / 4);
   //(8*3) This division by 4 might not be what's intended for sizeof(p1) / sizeof(struct p)
   if (x == sizeof(int) + 2 * sizeof(char))
      printf("True");
   else
      printf("False");
   return 0;
}
```

Output:

True

—----------

What would be the equivalent pointer expression for referring to the array element a[i][j][k][l]?

a) (((*(a+i)+j)+k)+l)

b) *(*(*(*(a+i)+j)+k)+l)

c) (*(*(a+i)+j)+k+l)

d) *((a+i)+j+k+l)

Output:

b

—----------

Damodar Kammili

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

Damodar Kammili

—----------

----

searching,sorting

Damodar Kammili

```
#include <stdio.h>

int main() {
    int data[ ] = {10, 20, 30};
    int target = 25;
    int result = -1;
    for (int i = 0; i < 3; i++) {
        if (data[i] == target) {
            result = i;
            break;
        }
    }
    printf("%d\n", result);
    return 0;
}
```

Answer the following questions based on the above program

1) What will be the exact output when this program is compiled and run?

Solution : -1

---

2) What is the primary purpose of the break statement in this specific linear search implementation?

a) **To exit the if condition.**

b) **To terminate the entire main function.**

c) **To stop the for loop as soon as the key is found.**

d) **To skip the current iteration and move to the next.**

Solution : C

---

3) If you wanted to modify this code to find the *last* occurrence of the key (assuming the key might appear multiple times, e.g., {5, 9, 1, 9, 3} and key = 9), which of the following changes would be most appropriate *without* changing the loop direction?

a) **Change found_idx = i; to found_idx = i + 1;**

b) **Remove the break statement in the program.**

c) Change int i = 0; to int i = 3; and i < 4 to i >= 0.

d) Change if (arr[i] == key) to if (arr[i] != key)

Solution : b

—----------

4) Consider the array arr[] = {5, 1, 9, 3}. If key were changed to 5, how many comparisons (arr[i] == key) would be made in the for loop before the key is found and the loop terminates?

Solution : 1 comparison

—----------

5) What if our **target is** 0 then what is the value of **result** variable ?

```
int data[ 4 ] = { 5, 1, 9 } ;
 target = 0;
  int result = -1;
   for (int i = 0; i < 4; i++) {
      if (data[i] == target) {
         result = i;
```

Solution : 3

—----------

Binary Search

—----------

```
#include <stdio.h>
int main() {
   int arr[] = {2, 4, 6, 8, 10};
   int low = 0, high = 4, key = 6, mid;
   int found_idx = -1;

   while (low <= high) {
      mid = low + (high - low) / 2; // Line B
      if (key == a[mid] ) {
```

```
            found_idx= mid;
            break;
        } else if (key > a[mid] ) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    printf("%d\n", found_idx);
    return 0;
}
```

---

1 . What is the primary reason for calculating `mid` using `low + (high - low) / 2` (at `// Line B`) instead of simply `(low + high) / 2`?

a) It is computationally faster.

b) It prevents integer overflow when `low` and `high` are very large.

c) It ensures `mid` is always an even number.

d) It handles negative indices correctly.

Solution : b

Reason: let us take an example like low limit =50 and end limit=100 then

*** (low+high)/2 => (50+100)/2 => 150/2 =75**

*** Low + (high-low)/2 => 50 +(100-50) /2 => 50+25 = 75**

In both cases the output is the same but the second one will ensure integer overflow.

---

2. In the `else if (arr[mid] < key)` block, why is `low` updated to `mid + 1` (and not just `mid`)?

a) To search the left half of the current sub-array.

b) To prevent an infinite loop if the key is not found.

c) Because `arr[mid]` has already been checked and is not the `key`.

d) To always include `mid` in the next search range.

Solution: c

---

3. If the `key` in the program were changed from `6` to `7`, what would be the output?

Solution : **-1**

Reason : `found_idx` remains its initial value of `-1`.

—----------

—----------

—----------

—----------

Sorting

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

Macros

```
#include <stdio.h>

#define func1(a,b) (a) > (b) ? (a) : (b)
#define func2(a,b) {int temp=a; a=b; b=temp ;}

int main()
{
   int a=3,b=5;
   if((3+func1(a,b)) > b)
   {
      func2(a,b);
   }
   printf("%d %d",a,b);
   return 0;
}
```

Output:

5 3

```
#include <stdio.h>

// The preprocessor directive #if A == 1 checks if A is defined and if its value is 1.
// However, 'A' is not defined anywhere in this code snippet.
// When an identifier like 'A' is used in an #if or #elif directive without being defined (e.g., using #define A 1),
// it is treated as having a value of 0.
// Therefore, A == 1 evaluates to 0 == 1, which is false.

#if A == 1
   #define B 0
#else
   #define B 1  // This block will be executed because A is treated as 0, so A == 1 is false.
#endif

int main()
{
```

```
    printf("%d", B); // B will be replaced by its defined value, which is 1.
    return 0;
}
```

Output:

1

—----------

What will be the output ? it contains error then write "Compilation error"

```
#include <stdio.h>

#define a 10

int main()
{
    printf("%d ",a);
    int a=50;
    printf("%d ",a);
    return 0;
}
```

Output:

Compilation error

—----------

In the C programming language, #define is a **preprocessor directive** used to define **macros**. It tells the compiler to replace all instances of a particular name in the code with a specific value or piece of code **before actual compilation begins**.

```
#include<stdio.h>
#define fun(x) (x*x)

int main() {
    float i;
    i = 64.0 / fun(2);
```

```
    printf("%.2f", i);
    return 0;
}
```

output:

16.00

Damodar Kammili

—----------

incre/decrement

```
#include <stdio.h>
int main()
{
    int i = 0, j = 1;
    printf("\n %d", i++ && ++j);          // Think about the rule of && evaluation
    printf("\n %d %d", i, j);
    return 0;
}
```

Output :

```
0
1 1
```

```
#include <stdio.h>

int main()
{
    int var1 = 10, var2 = 6;
    if (var1 == 5)
    {
        var2++;
    }
    printf("%d %d", var1, var2++);
    return 0;
}
```

Output:

```
5 7
```

```
#include <stdio.h>
int main()
{
    if( 'A'< 'a' )
    printf("oh you are shorter ah?");
    else
    printf("you are big na..");
    return 0;
}
```

Output:

```
 Oh you are shorter ah?
```

Damodar Kammili

Damodar Kammili

Structures

Damodar Kammili

```
#include <stdio.h>

int main()
{
    struct xyz {
        int a;
    };

    struct xyz obj1 = {11};
    struct xyz obj2 = obj1;

    printf("%d ", obj2.a);
    obj2.a = 101;
    printf("%d ", obj1.a);
    printf("%d ", obj2.a);

    return 0;
}
```

Output :

11 11 **101**

// Padding

What is the output of the following code?

```
typedef struct node
{
    int a;
    char c;
} n;

int main()
{
    printf("%lu ", sizeof(n));
    return 0;
}
```

Output:

8

// what will happen when we do like below?

```
Struct padding{
char z;
int a;
char c;
}n;
```

Output:

```
12
-----------------------------------------------


-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
------

-----------------------------------------------------------------
--

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------


-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------

-----------------------------------------------------------------
-------------------------------------------------
```

Damodar Kammili

Patterns

Damodar Kammili

Programs on Patterns

// print Right angled triangle

```
#include <stdio.h>

int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

// printing a Square

```
#include <stdio.h>

int main() {
    int rows = 4;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < rows; j++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

// Print Pyramid

```
#include <stdio.h>
int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= rows - i; j++) {
            printf(" ");
        }
        for (int k = 1; k <= (2 * i - 1); k++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Output:

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

// Print an inverted triangle or pyramid

```
#include <stdio.h>

int main() {
    int rows = 5;

    // Outer loop for rows (from top to bottom of inverted
pyramid)
    for (int i = rows; i >= 1; i--) {
        // Loop for printing leading spaces
        // The number of spaces increases as 'i' decreases
        for (int j = 1; j <= rows - i; j++) {
            printf(" ");
        }
    // the above code prints right angled spaces.
        // Loop for printing stars
        // The number of stars decreases as 'i' decreases
        for (int k = 1; k <= (2 * i - 1); k++) {
            printf("*");
        }
```

```
        printf("\n"); // Move to the next line after each row
    }

    return 0;
}
```

Output:

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

// Print Numbers in a triangle

```
#include <stdio.h>

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

Output:

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

Damodar Kammili

Basic level

```
#include <stdio.h>

int main() {
    int n=5;

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

    return 0;
}
```

Output:

```
1 2 3 4 5
```

```
#include <stdio.h>

int main() {
    int n=5, sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
   printf("Sum of first %d natural numbers is: %d\n", n, sum);

    return 0;
}
```

Output:

```
Sum of first 5 natural numbers is: 15
```

// printing Even Numbers

```
#include <stdio.h>
int main() {
    int n=10;

    printf("Even numbers from 1 to %d are: ", n);
    for (int i = 2; i <= n; i += 2) {
        printf("%d ", i);
    }

    return 0;
}
```

Output:

```
Even numbers from 1 to 10 are : 2 4 6 8 10
```

// printing Odd numbers

```
#include <stdio.h>
int main() {
    int n = 10 ;
printf("Odd numbers from 1 to %d are: ", n);
    for (int i = 1; i <= n; i += 2) {
        printf("%d ", i);
    }
return 0;
}
```

Output:

```
Odd numbers from 1 to 10 are: 1 3 5 7 9
```

// Square of a number

```
#include <stdio.h>

int main() {
    int n = 5 ;
    printf("Squares of numbers from 1 to %d are:\n", n);
    for (int i = 1; i <= n; i++) {
        printf("%d^2 = %d\n", i, i * i);
    }

    return 0;
}
```

Output:

```
Squares of numbers from 1 to 5 are:
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
5^2 = 25
```

// print sum of even numbers

```
#include <stdio.h>

int main() {
    int n, sum = 0;

    printf("Enter a number: ");
```

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

    for (int i = 2; i <= n; i += 2) {
        sum += i;
    }

    printf("Sum of even numbers up to %d is: %d\n", n, sum);

    return 0;
}
```

```
// Numbers divisible by 3
#include <stdio.h>
int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Numbers divisible by 3 up to %d are:\n", n);



for (int i = 1; i <= n; i++) {
        if (i % 3 == 0) {
           printf("%d ", i);
        }
    }
    return 0;
}
```

```
// 5 Multiples
#include <stdio.h>

int main() {
    int n;

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

    printf("Multiples of 5 up to %d are:\n", n);
    for (int i = 5; i <= n; i += 5) {
        printf("%d ", i);
    }

    return 0;
```

```
}
```

```
// Counting digits in a number
#include <stdio.h>
int main() {
    int n, count = 0;
    printf("Enter a number: ");
    scanf("%d", &n);
    for (int temp = n; temp != 0; temp /= 10) {
        count++;
    }
    printf("The number of digits in %d is: %d\n", n, count);

    return 0;
}
```

Damodar Kammili

NPTEL programs

Week-04: Programs(4)

Week-04:

Program-01

Write a C Program to Find the Smallest Number among Three Numbers (integer values) using Nested IF-Else statement.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	`90 -9 -8`	`-9 is the smallest number.`	`-9 is the smallest number.`	`Passed`
Test Case 2	`100 200 400`	`100 is the smallest number.`	`100 is the smallest number.`	`Passed`

Program-02

The length of three sides are taken as input. Write a C program to find whether a triangle can be formed or not. If not display “This Triangle is NOT possible.” If the triangle can be formed then check whether the triangle formed is equilateral, isosceles, scalene or a right-angled triangle. (If it is a right-angled triangle then only print Right-angle triangle do not print it as Scalene Triangle).

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	`5 12 13`	`Right-angle Triangle`	`Right-angle Triangle`	`Passed`
Test Case 2	`9 9 9`	`Equilateral Triangle`	`Equilateral Triangle`	`Passed`

Program-03

Write a program to find the factorial of a given number using while loop.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	`7`	`The Factorial of 7 is : 5040`	`The Factorial of 7 is : 5040`	`Passed`
Test Case 2	`11`	`The Factorial of 11 is : 39916800`	`The Factorial of 11 is : 39916800`	`Passed`

Program-04

Write a Program to find the sum of all even numbers from 1 to N where the value of N is taken as input. [For example when N is 10 then the sum is 2+4+6+8+10 = 30]

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	15	Sum = 56	Sum = 56	Passed
Test Case 2	25	Sum = 156	Sum = 156	Passed

Week-04: Program-01

Write a C Program to Find the Smallest Number among Three Numbers (integer values) using Nested IF-Else statement.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	90 -9 -8	-9 is the smallest number.	-9 is the smallest number.	Passed
Test Case 2	100 200 400	100 is the smallest number.	100 is the smallest number.	Passed

:

```c
#include <stdio.h>
int main()
{
   int n1, n2, n3;
if(n1<n2 && n2<n3)
 printf("%d is the smallest number.", n1);
else
 if(n2<n3)
 printf("%d is the smallest number.", n2);
 else
 printf("%d is the smallest number.", n3);
}
```

ANother solution:

Sample solutions (Provided by instructor)

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

   int n1, n2, n3;
   scanf("%d %d %d", &n1, &n2, &n3);
if (n1<n2)

   {

       if(n1<n3)

           printf("%d is the smallest number.", n1);

       else

           printf("%d is the smallest number.", n3);

   }

   else
   {

       if(n2<n3)

           printf("%d is the smallest number.", n2);

       else

           printf("%d is the smallest number.", n3);

   }
}
```

Week-04: Program-02

The length of three sides are taken as input. Write a C program to find whether a triangle can be formed or not. If not display “This Triangle is NOT possible.” If the triangle can be formed then check whether the triangle formed is equilateral, isosceles, scalene or a right-angled triangle. (If it is a right-angled triangle then only print Right-angle triangle do not print it as Scalene Triangle).

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	`5 12 13`	`Right-angle Triangle`	`Right-angle Triangle`	`Passed`
Test Case 2	`9 9 9`	`Equilateral Triangle`	`Equilateral Triangle`	`Passed`

```c
#include<stdio.h>
int main()

{

   int a,b,c;
   scanf("%d %d %d",&a, &b, &c); 16
if(a+b >c && b+c >a && c+a > b)

{

 if(a*a + b*b == c*c)

   printf("Right-angle Triangle");

 else

   if(a!=b && b!=c && c!=a)

   printf("Scalene Triangle");

   else

     if( a==b && b==c )

       printf("Equilateral Triangle");

     else
        if( a==b || b==c )

        printf("Isosceles Triangle");

}
```

```
else

 printf("Triangle is not possible");


}
```

Sample solutions (Provided by instructor)

```
#include<stdio.h>

int main()
{

   int a,b,c;

   scanf("%d %d %d",&a, &b, &c);
if(a<(b+c)&&b<(a+c)&&c<(a+b))
   {
       if(a==b&&a==c&&b==c)

       printf("Equilateral Triangle");
         else if(a==b||a==c||b==c)

         printf("Isosceles Triangle");

         else

   if((a*a)==(b*b)+(c*c)||(b*b)==(a*a)+(c*c)||(c*c)==(a*a)+(b*b))

       printf("Right-angle Triangle");

       else if(a!=b&&a!=c&&b!=c)

       printf("Scalene Triangle");

   }

   else

   printf("Triangle is not possible");
}
```

Week-04 Program-03

Write a program to find the factorial of a given number using while loop.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	7	The Factorial of 7 is : 5040	The Factorial of 7 is : 5040	Passed
Test Case 2	11	The Factorial of 11 is : 39916800	The Factorial of 11 is : 39916800	Passed

```
#include<stdio.h>
void main()

{   int n;
   long int fact;  /* n is the number whose factorial we have to find and fact is the factorial */
   scanf("%d",&n);  /* The value of n is taken from test cases */



fact=1;

int i;

/*

// Another solution using while loop

while(n>=1)
{

 fact=fact*n;
 n--;

}

printf("The Factorial of %d is : %ld",m,fact);



}

*/



for(i=1; i<=n; i++)
```

```
 fact=fact*i;


printf("The Factorial of %d is : %ld",n,fact);

}
```

Sample solutions (Provided by instructor)

```
#include<stdio.h>
void main()

{

   int n;

   long int fact;  /* n is the number whose factorial we have to find and
fact is the factorial */

   scanf("%d",&n);  /* The value of n is taken from test cases */



int i=1;

fact = 1;

while(i<=n)

   {

       fact*=i;

       i++;

   }

   printf("The Factorial of %d is : %ld",n,fact);

}
```

Week-04: Program-04

Write a Program to find the sum of all even numbers from 1 to N where the value of N is taken as input. [For example when N is 10 then the sum is 2+4+6+8+10 = 30]

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	15	Sum = 56	Sum = 56	Passed
Test Case 2	25	Sum = 156	Sum = 156	Passed

```c
#include <stdio.h>

void main()

{

int N, sum=0;

scanf("%d", &N);
int i;

for(i=2;i<=N;i+=2)

 sum=sum+i;

printf("Sum = %d", sum);

}
```

Sample solutions (Provided by instructor)

```c
#include <stdio.h>

void main()
 {

int N, sum=0;
 scanf("%d", &N);
  int i;

for (i = 1; i <= N; i++)

{

if (i % 2 == 0)

sum = sum + i;

}

printf("Sum = %d", sum);

}
```

Week-05 programs(4)

Week-05 Problem-01

Write a C program to check whether a given number (N) is a perfect number or not?

[Perfect Number - A perfect number is a positive integer number which is equals to the sum of its proper positive divisors. For example 6 is a perfect number because its proper divisors are 1, 2, 3 and it's sum is equals to 6**.]**

```
#include <stdio.h>

int main()

{

   int N;

   scanf("%d",&N);
int sum=0,i;

for(i=1;i<N;i++)

{

 if(N%i==0)

 sum=sum+i;

}

if(sum==N)

 printf("%d is a perfect number.",N);

else

 printf("%d is not a perfect number.",N);



}
```

Sample solutions (Provided by instructor)

```
#include <stdio.h>

int main()

{

   int N;

   scanf("%d",&N); 13
int i, sum=0;

   for(i=1; i<N;i++)

   {

       if(N%i==0)
```

```
            sum+=i;
    }

    if(sum==N)
        printf("\n%d is a perfect number.",N);
    else
        printf("\n%d is not a perfect number.",N);
}
```

Week-05 Problem-02

Write a C program to count total number of digits of an Integer number (N).

```
#include <stdio.h>

int main()

{
    int N;

    scanf("%d",&N);int c=0;

int m=N;


while(N>0)

{

 N=N/10;

 c++;

}
```

```
printf("The number %d contains %d digits.",m,c);


}
```

Week-05 Problem-03

Write a C program to check whether the given number(N) can be expressed as Power of Two (2) or not.
For example 8 can be expressed as 2^3.

```
#include <stdio.h>

int main()

{

   int N;

   scanf("%d",&N);
int flag=0;
int m=N;


if(N==1)

   printf("%d is a number that can be expressed as power of 2.",m);

else
```

```
{

while(N!=1)

{

 if(N%2==0)

 {

   N=N/2;

   //c++;    //We are counting the no of times that a number is divided with
2.
 }

 else

 {

   flag=1;

   break;

 }

}


if(flag==0)

 printf("%d is a number that can be expressed as power of 2.",m);

else

 printf("%d cannot be expressed as power of 2.",m);

}


}
```

Sample solution_ANOTHER WAY (Provided by instructor)

```
#include <stdio.h>


int main()


{

   int N;


   scanf("%d",&N);

int temp, flag;
```

```
    temp=N;

    flag=0;
    while(temp!=1)
    {   if(temp%2!=0){
            flag=1;
            break;
        }
        temp=temp/2;
    }

    if(flag==0)
        printf("%d is a number that can be expressed as power of 2.",N);
    else
        printf("%d cannot be expressed as power of 2.",N);
}
```

Week-05 Program-04

Write a C program to print the following Pyramid pattern upto Nth row. Where N (number of rows to be printed) is taken as input.

For example when the value of N is 5 the pyramid will be printed as follows

*****

****

***

**

*

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

{
```

```
int N;

scanf("%d", &N); /*The value of N is taken as input from the test case */

int i;
while(N>=1)

{

   for(i=1; i<=N; i++)

   printf("*");


   printf("\n");

   N--;

}


}
```

Sample solutions (Provided by instructor)

```
#include<stdio.h>

int main()

{
int N;

scanf("%d", &N); /*The value of N is taken as input from the test case */

int i,j;

for(i=N; i>0; i--)

 {

 for(j=0;j<i;j++)

   {

   printf("*");
   }

 printf("\n");
 }
}
```

Week-05

Problem-01

Write a C program to check whether a given number (N) is a perfect number or not?

[Perfect Number - A perfect number is a positive integer number which is equals to the sum of its proper positive divisors. For example 6 is a perfect number because its proper divisors are 1, 2, 3 and it's sum is equals to 6**.]**

Problem-02

Write a C program to count total number of digits of an Integer number (N).

Problem-03

Write a C program to check whether the given number(N) can be expressed as Power of Two (2) or not.

For example 8 can be expressed as 2^3.

Program-04

Write a C program to print the following Pyramid pattern upto Nth row. Where N (number of rows to be printed) is taken as input.

For example when the value of N is 5 the pyramid will be printed as follows

*****

****

***

**

*

Week-06 Program-(4)

Week-06 Program-01

Write a C Program to find Largest Element of an Integer Array.
Here the number of elements in the array ‘n’ and the elements of the array is read from the test data.

```
#include <stdio.h>


int main()

{

   int i, n, largest;
   int arr[100];



   scanf("%d", &n); /*Accepts total number of elements from the test data */



for(i = 0; i < n; ++i)
   {

      scanf("%d", &arr[i]); /* Accepts the array element from test data */

   }



largest=-1000;

for(i=0;i<n;i++)

{

 if(arr[i]>largest)

   largest=arr[i];

}

printf("Largest element = %d", largest);
}
```

Week-06 Program-02

Write a C Program to print the array elements in reverse order (Not reverse sorted order. Just the last element will become first element, second last element will become second element and so on)
Here the size of the array, ‘n’ and the array elements is accepted from the test case data. The last part i.e. printing the array is also written.
You have to complete the program so that it prints in the reverse order.

```
#include<stdio.h>

int main() {
  int arr[20], i, n;

  scanf("%d", &n); /* Accepts the number of elements in the array */


 for (i = 0; i < n; i++)
    scanf("%d", &arr[i]); /*Accepts the elements of the array */
int temp=0,j;
for(i=n-1,j=0; i>=n/2; i--,j++)
{
 temp=arr[i];

 arr[i]=arr[j];

 arr[j]=temp;

}



for (i = 0; i < n; i++) {
     printf("%d\n", arr[i]); // For printing the array elements
  }

  return (0);
}
```

Week-06 Program-03

Write a C program to read Two One Dimensional Arrays of same data type (integer type) and merge them into another One Dimensional Array of same type.

```c
#include<stdio.h>

int main()

{

  int arr1[20], arr2[20], array_new[40], n1, n2, size, i;

/*n1 size of first array (i.e. arr1[]), n2 size of second array(i.e. arr2[]),

  size is the total size of the new array (array_new[]) */


  scanf("%d", &n1); //Get the size of first array from test data and store it in n1.



  for (i = 0; i < n1; i++)

     scanf("%d", &arr1[i]); //Accepts the values for first array

  scanf("%d", &n2); //Get the size of second array from test data and store it in n2.



  for (i = 0; i < n2; i++)

     scanf("%d", &arr2[i]); //Accepts the values for second array




size=0;

for(i=0; i<n1; i++,size++)

  array_new[size]=arr1[i];



for(i=0; i<n2; i++, size++)

   array_new[size]=arr2[i];
```

```
// Another form of logic to the above program

/*
int j;


for(i=0; i<n1; i++)
  array_new[i]=arr1[i];



for(j=0;j<n2;j++,i++)
   array_new[i]=arr2[j];


size=i;

*/


//Printing after merging


for (i = 0; i < size; i++) {
     printf("%d\n", array_new[i]);
  }


}
```

Week-06 Program-04

Write a C Program to delete duplicate elements from an array of integers.

```
#include<stdio.h>

int main()

{

  int array[50], i, size;

  scanf("%d", &size); /*Accepts the size of array from test case data */

  for (i = 0; i < size; i++)

  scanf("%d", &array[i]); /* Read the array elements from the test case data
*/
```

```
int j=0,k=0;

for ( i = 0; i < size; i ++)

   {

       for ( j = i + 1; j < size; j++)

       {

           // use if statement to check duplicate element

           if ( array[i] == array[j])

           {

               // delete the current position of the duplicate element

               for ( k = j; k < size - 1; k++)

               {

                   array[k] = array[k + 1];

               }

               // decrease the size of array after removing duplicate element

               size--;


           // if the position of the elements is changes, don't increase the
index j
```

```
                j--;
            }
        }
    }



/*
int j=0;
for(i=0;i<size-2;i++)

for(i=j+1;i<size-1;i++)
{
 if(array[i]==array[j])
 {

   array[j]=array[j+1];
   j--;

   break;
 }
}


size=size-1;
*/

for (i = 0; i < size; i++) {
     printf("%d\n", array[i]);
  }

}
```

Week-06

Program-01

Write a C Program to find Largest Element of an Integer Array.
Here the number of elements in the array ‘n’ and the elements of the array is read from the test data.

Program-02

Write a C Program to print the array elements in reverse order (Not reverse sorted order. Just the last element will become first element, second last element will become second element and so on)
Here the size of the array, ‘n’ and the array elements is accepted from the test case data. The last part i.e. printing the array is also written.
You have to complete the program so that it prints in the reverse order.

Program-03

Write a C program to read Two One Dimensional Arrays of same data type (integer type) and merge them into another One Dimensional Array of same type.

Program-04

Write a C Program to delete duplicate elements from an array of integers.

Week-07 Programs(4)

Week-07

Program-01

Write a C Program to Count Number of Uppercase and Lowercase Letters in a given string. The given string may be a word or a sentence.

```c
#include<stdio.h>

int main() {

  int upper = 0, lower = 0;

  char ch[100];

  scanf(" %[^\n]s", ch);  /*A word or a sentence is accepted from test case data */



/* Complete the remaining part of the code to store number of uppercase letters

in the variable upper and lowercase letters in variable lower.

The print part of already written. You can declare any variable if necessary */

int i=0;


for(i=0;ch[i]!='\0';i++)

{

 if(ch[i]>='A'  && ch[i]<='Z')

   upper++;

 if(ch[i]>='a'  && ch[i]<='z')

   lower++;

}



printf("Uppercase Letters : %d\n", upper); /*prints number of uppercase letters */

  printf("Lowercase Letters : %d", lower); /*prints number of lowercase letters */


  return (0);

```

```
}
```

Program-02

Write a C program to find the sum of all elements of each row of a matrix.
Example: For a matrix 4 5 6
6 7 3
1 2 3

The output will be
15
16
6

```
#include <stdio.h>

int main()

{
   int matrix[20][20];

   int i,j,r,c;


   scanf("%d",&r); //Accepts number of rows

   scanf("%d",&c); //Accepts number of columns


   for(i=0;i< r;i++) //Accepts the matrix elements from the test case data

   {

       for(j=0;j< c;j++)
       {

           scanf("%d",&matrix[i][j]);
       }

int sum=0;


for(i=0;i< r;i++)

   {

       sum=0;


       for(j=0;j< c;j++)

       {

           sum = sum + matrix[i][j];
```

```
            }
            printf("%d\n",sum);
        }

    }
```

Program-03

Write a C program to find subtraction of two matrices i.e. matrix_A - matrix_B=matrix_C.

If the given martix are

2 3 5 and 1 5 2 Then the output will be 1 -2 3

4 5 6 2 3 4 2 2 2

6 5 7 3 3 4 3 2 3

The elements of the output matrix are separated by one blank space

```
#include <stdio.h>

int main()

{

   int matrix_A[20][20], matrix_B[20][20], matrix_C[20][20];

   int i,j,row,col;

   scanf("%d",&row); //Accepts number of rows

   scanf("%d",&col); //Accepts number of columns


   /* Elements of first matrix are accepted from test data */

   for(i=0; i<row; i++)

   {

       for(j=0; j<col; j++)

       {

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

       }

   }


    /* Elements of second matrix are accepted from test data */


   for(i=0; i<row; i++)

   {

       for(j=0; j<col; j++)

       {

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

       }

   }


for(i=0;i< row;i++)

   {

       //matrix_C[i][j]=0;

       for(j=0;j< col;j++)
```

```
        {
            matrix_C[i][j]=matrix_A[i][j]-matrix_B[i][j];

            printf("%d ",matrix_C[i][j]);
        }
        printf("\n");

    }

/*
// printing the matrix_C separately.

for(i=0; i<row; i++)
    {
        for(j=0; j<col; j++)
        {
            printf("%d ", matrix_C[i][j]);
        }
         printf("\n");

    }

*/
                                        }
```

Program-04

Write a C program to print lower triangle of a square matrix.
For example the output of a given matrix

2 3 4	will be	2 0 0
5 6 7		5 6 0
4 5 6		4 5 6

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

int matrix[20][20];

int i,j,r;



scanf("%d", &r); //Accepts number of rows or columns



   for(i=0;i< r;i++) //Accepts the matrix elements from the test case data

   {

       for(j=0;j<r; j++)

       {

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

       }

   }


for(i=0;i< r;i++)

   {

      for(j=0;j<r; j++)

       {

         if(j>i)

           printf("0 ");

         else

           printf("%d ",matrix[i][j]);

       }

        printf("\n");

   }


}
```

WEEK-08 programs(4)

Week-08

Week 8: Programming Assignment 1

Write a C Program to find HCF of 4 given numbers using recursive function

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	50 455 60 200	The HCF is 5	The HCF is 5	Passed
Test Case 2	67 89 45 41	The HCF is 1	The HCF is 1	Passed

```c
#include<stdio.h>

int HCF(int, int); //You have to write this function which calculates the HCF.



int main()
{

  int a, b, c, d, result;
  scanf("%d %d %d %d", &a, &b, &c, &d); /* Takes 4 number as input from the test data */

  result = HCF(HCF(a, b), HCF(c,d));

  printf("The HCF is %d", result);

}



//Complete the rest of the program to calculate HCF

int HCF(int a, int b) {
   if (b == 0)

       return a;  // Base case: if b is 0, HCF is a

   else

       return HCF(b, a % b);  // Recursive case using Euclidean algorithm
}
```

Sample solutions (Provided by instructor)

```c
#include<stdio.h>

int HCF(int, int); //You have to write this function which calculates the HCF.
```

```c
int main()
{
  int a, b, c, d, result;

  scanf("%d %d %d %d", &a, &b, &c, &d); /* Takes 4 number as input from the test data */

  result = HCF(HCF(a, b), HCF(c,d));

  printf("The HCF is %d", result);

}


//Complete the rest of the program to calculate HCF

int HCF(int x, int y)

{
  while (x != y)

  {
     if (x > y)

      {

          return HCF(x - y, y);

      }

      else

      {

      return HCF(x, y - x);

      }

   }

   return x;
}
```

Week 8: Programming Assignment 2

Write a C Program to find power of a given number using recursion. The number and the power to be calculated is taken from test case

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	5 3	5^3 is 125	5^3 is 125	Passed
Test Case 2	16 3	16^3 is 4096	16^3 is 4096	Passed

```c
#include <stdio.h>
long power(int, int);
int main()
{
int pow, num;
long result;

scanf("%d", &num); //The number taken as input from test case data

scanf("%d", &pow); //The power is taken from the test case
result = power(num, pow);
printf("%d^%d is %ld", num, pow, result);
return 0;
}
long power(int num, int pow)
{
 // below are some other base conditions to accept the code.
 /*

 if(pow)
   pow--;
 else
   return 1;



 if(pow==1)
   return num;
 else
   pow--;


*/

 if(pow==0)
```

```
   return 1;
 else
   pow--;

 return num * power(num, pow);
}
```

Another Sample solution (Provided by instructor)

```
#include <stdio.h>
long power(int, int);
int main()
{
int pow, num;
long result;

scanf("%d", &num); //The number taken as input from test case data

scanf("%d", &pow); //The power is taken from the test case
result = power(num, pow);
printf("%d^%d is %ld", num, pow, result);
return 0;
}
long power(int num, int pow)
{
if (pow)
{
return (num * power(num, pow - 1));
}
return 1;
}
```

Week 8: Programming Assignment 3

Write a C Program to print Binary Equivalent of an Integer using Recursion

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	30	The binary equivalent of 30 is 11110	The binary equivalent of 30 is 11110\n	Passed
Test Case 2	111	The binary equivalent of 111 is 1101111	The binary equivalent of 111 is 1101111\n	Passed

```c
#include <stdio.h>
int binary_conversion(int); //function to convert binary to decimal number
int main()
 {
 int num, bin;  //num is the decimal number and bin is the binary equivalent for the number


 scanf("%d", &num); //The decimal number is taken from the test case data
 bin = binary_conversion(num); //binary number is stored in variable bin
 printf("The binary equivalent of %d is %d\n", num, bin);
 return 0;
 }
int binary_conversion(int n) // we can also take num
{
 if (n == 0)
       return 0;
   else
       return (n % 2) + 10 * binary_conversion (n / 2);  // Recursive formula
}
```

Another Sample solutions (Provided by instructor)

```c
#include <stdio.h>
int binary_conversion(int); //function to convert binary to decimal number
int main()
```

```
{
 int num, bin;  //num is the decimal number and bin is the binary equivalent for the number

 scanf("%d", &num); //The decimal number is taken from the test case data
 bin = binary_conversion(num); //binary number is stored in variable bin
 printf("The binary equivalent of %d is %d\n", num, bin);
 return 0;
 }
int binary_conversion(int num)
 {
    if (num == 0)
        {
            return 0;
        }
       else
        {
            return (num % 2) + 10 * binary_conversion(num / 2);
        }
 }
```

Week 8: Programming Assignment 4

Write a C Program to reverse a given word using function. e.g. INDIA should be printed as AIDNI

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	INDIA	The string after reversing is: AIDNI	\n The string after reversing is: AIDNI	Passed
Test Case 2	DELHI	The string after reversing is: IHLED	\n The string after reversing is: IHLED	Passed

```
#include<stdio.h>
```

```c
#include<string.h>

void reverse(char[], int, int);
   int main()
   {
        char str1[20];
        scanf("%s", str1); //The string is taken as input form the test data.


//int length;
//length=strlen(str1)-1;


reverse(str1,0,strlen(str1)-1);


}

void reverse(char str2[], int i, int length)
{
 int temp,j;

 for(i=0,j=length; i< j ; j--,i++ )    //we can also write j>length/2
 {
   temp=str2[i];
   str2[i]=str2[j];
   str2[j]=temp;
 }
 printf("\nThe string after reversing is: %s", str2);
}



/*
// another form of solution


reverse(str1,0,strlen(str1)-1);
```

```
}

void reverse(char str[], int start, int end) {
   while (start < end) {
        char temp = str[start];
        str[start] = str[end];
        str[end] = temp;
        start++;
        end--;
   }

*/
```

Another Sample solution(Provided by instructor)

```
#include<stdio.h>
#include<string.h>

void reverse(char[], int, int);
   int main()
   {
        char str1[20];
        scanf("%s", str1); //The string is taken as input form the test data.

/* Complete the program to print the string in reverse order using the
function
void reverse(char[], int, int);
Use the printf statement as
printf("The string after reversing is: %s\n", str1);
You can use different variable also.
*/
int size;
   size = strlen(str1);
        reverse(str1, 0, size - 1);
```

```
    printf("The string after reversing is: %s", str1);
    return 0;
}


void reverse(char str1[], int index, int size)
{
    char temp;
    temp = str1[index];
    str1[index] = str1[size - index];
    str1[size - index] = temp;
    if (index == size / 2)
    {
        return;
    }
    reverse(str1, index + 1, size);
}
```

Week -09 programs(4)

Week-09 Program-01

Write a program to print all the locations at which a particular element (taken as input) is found in a list and also print the total number of times it occurs in the list. The location starts from 1.

For example if there are 4 elements in the array
5
6
5
7
If the element to search is 5 then the output will be
5 is present at location 1
5 is present at location 3
5 is present 2 times in the array.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	7 30 50 90 30 70 30 30 30	30 is present at location 1.\n 30 is present at location 4.\n 30 is present at location 6.\n 30 is present at location 7.\n 30 is present 4 times in the array.	30 is present at location 1.\n 30 is present at location 4.\n 30 is present at location 6.\n 30 is present at location 7.\n 30 is present 4 times in the array.\n	Passed
Test Case 2	4 50 60 20 10 80	80 is not present in the array.	80 is not present in the array.\n	Passed

```c
#include <stdio.h>

int main()

{

  int array[100], search, n, count = 0;

  //"search" is the key element to search and 'n' is the total number of element of the array

  // "count" is to store total number of elements
```

```
scanf("%d", &n); //Number of elements is taken from test case


int c;
  for (c = 0; c < n; c++)
     scanf("%d", &array[c]);


  scanf("%d", &search); // The element to search is taken from test case



for(c=0;c<n;c++)
{
 if( array[c]==search)
 {
     printf("%d is present at location %d.\n",search,c+1);
     count++;
 }
}


if(count)
printf("%d is present %d times in the array.\n",search,count);


else
   printf("%d is not present in the array.\n",search);
}
```

Another Sample solution (Provided by instructor)

```
#include <stdio.h>
int main()
{
  int array[100], search, n, count = 0;
  //"search" is the key element to search and 'n' is the total number of
element of the array
```

```
  // "count" is to store total number of elements

scanf("%d", &n); //Number of elements is taken from test case

int c;
  for (c = 0; c < n; c++)
     scanf("%d", &array[c]);

  scanf("%d", &search); // The element to search is taken from test case

for (c = 0; c < n; c++)
   {
     if (array[c] == search)
     {
        printf("%d is present at location %d.\n", search, c+1);
        count++;
     }
   }
  if (count == 0)
     printf("%d is not present in the array.\n", search);
  else
     printf("%d is present %d times in the array.\n", search, count);

  return 0;
}
```

Week-09 Program-02

Write a C program to search a given element from a 1D array and display the position at which it is found by using linear search function. The index location starts from 1.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	4 45 65 85 25 95	95 is not present in the array.	95 is not present in the array.\n	Passed
Test Case 2	5 6 9 5 4 7 6	6 is present at location 1.	6 is present at location 1.\n	Passed

```c
#include <stdio.h>

int linear_search(int[], int, int);

int main()

{

  int array[100], search, c, n, position;

  /* search - element to search, c - counter, n - number of elements in array,

  position - The position in which the element is first found in the list. */


   scanf("%d", &n); // Number of elements in the array is read from the test case data


   for (c = 0; c < n; c++)

   scanf("%d", &array[c]); //Elements of array is read from the test data


```

```
   scanf("%d", &search);  //Element to search is read from the test case data
15
16
position=linear_search(array,n,search);
21
if(position==-1)
22
    printf("%d is not present in the array.\n", search);
23
else
24
25
printf("%d is present at location %d.\n", search, position+1); //As array[0]
has the position 1
26
//printf("%d is present at location %d.\n", search, position); //As we are
receiving c+1 then print like this.
27
28
29
}
30
31
32
33
int linear_search(int array[],int n,int search)
34
{
35
 int c;
36
  for(c=0;c<n;c++)
37
   {
38
    if(array[c]==search)
39
       return c;            // return c+1;
40
   }
41
42
return -1;
43
44
}
45
46
47
```

Sample solution (Provided by instructor)

```
1
#include <stdio.h>
2
int linear_search(int[], int, int);
3
int main()
4
```

```
{
  int array[100], search, c, n, position;
  /* search - element to search, c - counter, n - number of elements in
array,
  position - The position in which the element is first found in the list. */


   scanf("%d", &n); // Number of elements in the array is read from the test
case data


   for (c = 0; c < n; c++)
   scanf("%d", &array[c]); //Elements of array is read from the test data


   scanf("%d", &search);  //Element to search is read from the test case data


  /* Use the following in the printf statement to print the output
  printf("%d is not present in the array.\n", search);
  printf("%d is present at location %d.\n", search, position+1); //As
array[0] has the position 1
  */
position = linear_search(array, n, search);


  if (position == -1)
     printf("%d is not present in the array.\n", search);
  else
     printf("%d is present at location %d.\n", search, position+1);
  return 0;
}


int linear_search(int a[], int n, int find) {
  int c;
  for (c = 0 ;c < n ; c++ )
   {
     if (a[c] == find)
        return c;
```

```
    }
  return -1;
}
```

Week-09 Program-03

Write a C program to search a given number from a sorted 1D array and display the position at which it is found using binary search algorithm. The index location starts from 1.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	6 1 2 3 4 5 6 2	2 found at location 2.	2 found at location 2.\n	Passed
Test Case 2	7 40 50 60 70 80 90 100 100	100 found at location 7.	100 found at location 7.\n	Passed

```c
#include <stdio.h>
int main()
{
int c, n, search,
array[100];
scanf("%d",&n); //number of elements in the array

for (c = 0; c < n; c++)
scanf("%d",&array[c]);

scanf("%d", &search); //The element to search is read from test case.
```

```
int binarySearch(int [], int , int);
int result;



result = binarySearch(array,n,search);
if(result == -1)
  printf("Not found! %d isn't present in the list.\n", search);
else
  printf("%d found at location %d.\n", search, result);
}



int binarySearch(int arr[], int size, int target) {
   int left = 0, right = size - 1, mid;

   while (left <= right) {
       mid = (right +left) / 2; // Calculate the middle index

       if (arr[mid] == target)
           return mid + 1; // Return 1-based index

       if (arr[mid] < target)
           left = mid + 1; // Search in right half
       else
           right = mid - 1; // Search in left half
   }

   return -1; // Element not found
}
```

```
49
```

Sample solutions (Provided by instructor)

```
1
#include <stdio.h>
2
int main()
3
{
4
int c, n, search,
5
array[100];
6
scanf("%d",&n); //number of elements in the array
7

8
for (c = 0; c < n; c++)
9
scanf("%d",&array[c]);
10

11
scanf("%d", &search); //The element to search is read from test case.
int first, last, middle;
18
  first = 0;
  last = n - 1;
20
  middle = (first+last)/2;

  while (first <= last) {
23
     if (array[middle] < search)
24
        first = middle + 1;
25
     else if (array[middle] == search) {
26
        printf("%d found at location %d.\n", search, middle+1);
27
        break;
28
     }
29
     else
30
        last = middle - 1;
31

32
     middle = (first + last)/2;
33
     }
34
  if (first > last)
35
```

```
     printf("Not found! %d isn't present in the list.\n", search);

    return 0;
     }
```

Week-09 Program-04

Write a C program to reverse an array by swapping the elements and without using any new array.

Private Test cases used for evaluation	Input	Expected Output	Actual Output	Status
Test Case 1	7 8 9 10 6 4 7 11	Reversed array elements are:\n 11\n 7\n 4\n 6\n 10\n 9\n 8	Reversed array elements are:\n 11\n 7\n 4\n 6\n 10\n 9\n 8\n	Passed

```
#include <stdio.h>

int main() {

 int array[100], n, c;

 scanf("%d", &n); // n is number of elements in the array.

 for (c = 0; c < n; c++) {

   scanf("%d", &array[c]);

 }

int temp,i,j;


for(i=0,j=n-1;i<n/2;i++,j--) // j>=n/2;

{

  temp    = array[i];


 array[i] = array[j];


 array[j] = temp;


}


printf("Reversed array elements are:\n");
```

```
 for (c = 0; c < n; c++) {
   printf("%d\n", array[c]);
 }
 return 0;
}
```

Sample solutions (Provided by instructor)

```
#include <stdio.h>
int main() {
 int array[100], n, c;
 scanf("%d", &n); // n is number of elements in the array.
 for (c = 0; c < n; c++) {
   scanf("%d", &array[c]);
 }
int temp, end;
  end = n - 1;
  for (c = 0; c < n/2; c++) {
    temp       = array[c];
    array[c]   = array[end];
    array[end] = temp;

    end--;
 }
printf("Reversed array elements are:\n");
 for (c = 0; c < n; c++) {
   printf("%d\n", array[c]);
 }
 return 0;
}
```

NPTEL QUIZ assignments - 2025

This tab consists of the list of subtabs each consists of every week **Quiz** questions.

Week 1 : Assignment

Week 1 : Assignment 1

Set of instructions to be provided to an electronic machine to perform a task is called

a) Programming

b) Processing

c) Computing

d) Compiling

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) Programming

1 point

Compiler helps in the translation from

a) Integer to binary

b) High-level program to binary digits

c) High-level language to machine level language

d) Pseudo code to computer program

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) High-level language to machine level language

1 point

The ALU unit of computer

a) Can perform logical operation only

b) Can perform arithmetic operation only

c) Can perform both arithmetic and logical operations

d) None of the above.

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Can perform both arithmetic and logical operations

1 point

What type of device is computer printer?

a) Memory

b) Output

c) Storage

d) Input

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) Output

1 point

Algorithm is -

a) A process or set of rules to be followed in calculations or other problem-solving operations, especially by a human.

b) A process or set of rules to be followed to solve numerical problems only.

c) A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

d) A process or set of rules to be followed to solve logical problems only.

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

1 point

When we write X=10 and Y=X, which of the following memory assignment is correct

a) X and Y will have same location and 10 will be stored.

b) X and Y will have two distinct locations and 10 will be stored in both.

c) X and Y will have same location and only X will contain value 10

d) X and Y will have two distinct locations and only X will contain value 10

Yes, the answer is correct.

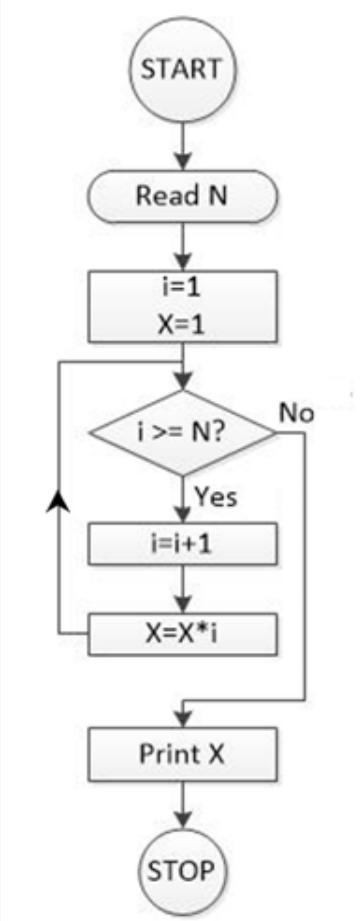
Score: 1

Accepted Answers:

b) X and Y will have two distinct locations and 10 will be stored in both.

1 point

The input N from the user is 6. The output of the following algorithm is



a) 21

b) 720

c) 1

d) 2

Yes, the answer is correct.

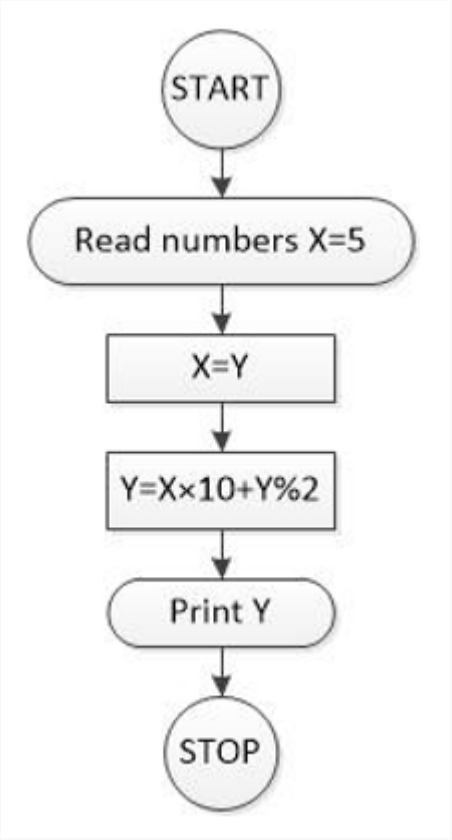
Score: 1

Accepted Answers:

c) 1

1 point

What will be the output of the algorithm given below?



a) 51

b) 52

c) 50

d) Compilation error

Yes, the answer is correct.

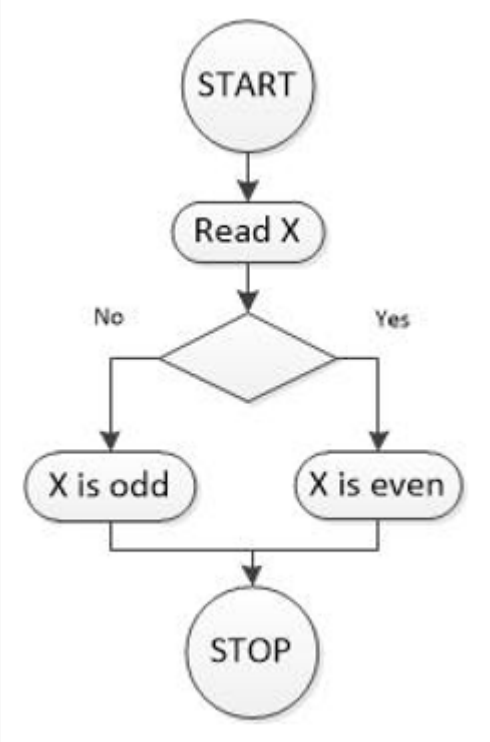
Score: 1

Accepted Answers:

d) Compilation error

1 point

The following algorithm is used to find a number X is even or odd. What will be the content of the empty box?



a) X%10=0?

b) X/10=0?

c) X/2=0?

d) X%2=0?

Yes, the answer is correct.

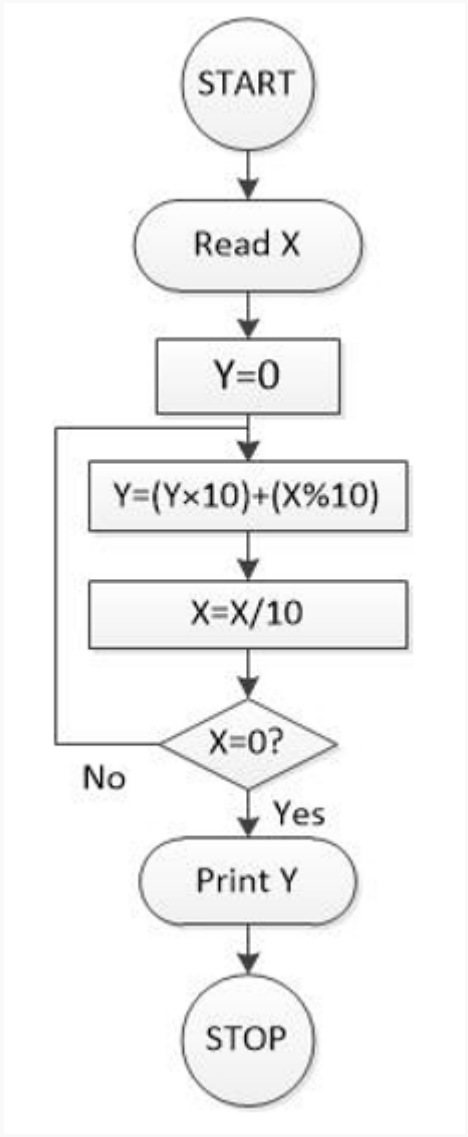
Score: 1

Accepted Answers:

d) X%2=0?

1 point

X is an integer (X=2648). The print value of Y of the flowchart below is



a) 20

b) 22664488

c) 8462

d) 0

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 8462

Week 2 : Assignment

Week 2 : Assignment 2

Which of the following cannot be used as a variable in C programming?

a) Var123

b) Var_123

c) 123Var

d) X_123_Var

No, the answer is incorrect.

Score: 0

Accepted Answers:

c) 123Var

1 point

Which of the following is not a correct variable type in C?

a) int

b) double

c) char

d) All of above are correct variable type

No, the answer is incorrect.

Score: 0

Accepted Answers:

d) All of above are correct variable type

1 point

The execution of any C program is

a) Sequential

b) Parallel

c) Multi-threading

d) None of these

No, the answer is incorrect.

Score: 0

Accepted Answers:

a) Sequential

1 point

Which of the following statements is correct?

I. Keywords are those words whose meaning is already defined by Compiler.

II. Keywords cannot be used as variable names.

III. There are 32 keywords in C.

IV. C keywords are also called as reserved words.

a) I and II

b) II and III

c) I, II and IV

d) All of the above

No, the answer is incorrect.

Score: 0

Accepted Answers:

d) All of the above

1 point

A function is

a) Block of statements to perform some specific task

b) It is a fundamental modular unit to perform some task

c) It has a name and can be used multiple times

d) All of the above

No, the answer is incorrect.

Score: 0

Accepted Answers:

d) All of the above

1 point

What will be the output? [N.B: - .2f is used to print up to 2 decimal places of a floating point number]

```
#include <stdio.h>
int main()
{
  float a = 5.0;
  printf ("The output is  %.2f", (9/5)*a + 7);
  return 0;
}
```

a) 28.2

b) 21.00

c) 16.00

d) 12.00

No, the answer is incorrect.

Score: 0

Accepted Answers:

d) 12.00

1 point

What is the output of the following C code?

```
#include <stdio.h>
int main()
{
        int var = 0110;
        var=var+7;
        printf("%d", var);
        return 0;
}
```

a) 106

b) 70

c) 79

d) Compiler error

No, the answer is incorrect.

Score: 0

Accepted Answers:

c) 79

1 point

If integer needs two bytes of storage, then the minimum value of a signed integer in C would be

a) $-(2^{16}-1)$

b) 0

c) $-(2^{15}-1)$

d) -2^{15}

No, the answer is incorrect.

Score: 0

Accepted Answers:

d) -2^{15}

1 point

What will be the output of the program given below?

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

a) 9

b) 0

c) 1001

d) Compilation Error

No, the answer is incorrect.

Score: 0

Accepted Answers:

d) Compilation Error

What is the output?

```
#include<stdio.h>
#define fun(x) (x*x-x)
int main()
{
  float i;
  i = 37.0/fun(2);
  printf("%.2f", i);
  return 0;
}
```

Hint

No, the answer is incorrect.

Score: 0

Accepted Answers:

(Type: Regex Match) 8.5

Week 3 : Assignment 3

Week 3 : Assignment 3

The precedence of arithmetic operators is (from highest to lowest)

a) %, *, /, +, –

b) %, +, /, *, –

c) +, -, %, *, /

d) %, +, -, *, /

Yes, the answer is correct.

Score: 1

Accepted Answers:

*a) %, *, /, +, –*

1 point

What is the output of the following program? (% indicates modulo operation, which results the reminder of a division operation)

```
#include <stdio.h>
int main()
 {
        float i = -3.0;
        int k = i % 2;
        printf("%d", k);
        return 0;
}
```

a) -1

b) 1

c) 0

d) Compilation error

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) Compilation error

1 point

Find the output of the following C code. (% indicates modulo operation, which results the reminder of a division operation)

```
#include<stdio.h>
int main()
{
        int a=10, b=3, c=2, d=4, result;
        result=a+a*-b/c%d+c*d;
        printf("%d", result);
        return 0;
}
```

a) -42

b) 24

c) 15

d) -34

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 15

1 point

What is the output of the following C code?

```
#include <stdio.h>
int main()
{
int h = 8;
int b = 4 * 6 + 3 * 4 < h*5 ?4 : 3;
printf("%d\n", b);
return 0;

}
```

a) 0

b) 3

c) 4

d) Compilation error

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 4

1 point

What will be the output?

```
#include <stdio.h>
int main ()
{
   int a = 1, b = 2, c = 3;
   if (c > b > a)
         printf("TRUE");
   else
         printf("FALSE");
   return 0;
}
```

a) TRUE

b) FALSE

c) Syntax Error

d) Compilation Error

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) FALSE

1 point

Find the output of the following C code

```
#include <stdio.h>
int main()
{
int x=1;
   if ((3>5) || (2!=3))
printf("IITKGP\n");
   else if (x&=0)
printf("IITD\n");
   else
printf("IITM\n");
return 0;
}
```

a) IITKGP

b) IITD and IITM

c) IITKGP and IITM

d) IITM

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) IITKGP

1 point

What will be the output?

```
#include <stdio.h>
int main()
{
int x=2;
   if(x=1)
printf("TRUE");
   else
printf("FALSE");
   return 0;
}
```

a) TRUE

b) FALSE

c) Compilation Error

d) Compiler Dependent

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) TRUE

1 point

Which of the following statement is correct?

a) Operator precedence determines which operator is performed first in an expression with more than one operator with different precedence.

Associativity is used when two operators of same precedence appear in an expression

b) Operator associativity determines which operator is performed first in an expression with more than one operator with different associativity.

Precedence is used when two operators of same precedence appear in an expression

c) Operator precedence and associativity are same.

d) None of the above

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) Operator precedence determines which operator is performed first in an expression with more than one operator with different precedence.

Associativity is used when two operators of same precedence appear in an expression

1 point

Which of the following method are accepted for assignment?

a) 8=x=y=z

b) x=8=y=z

c) x=y=z=8

d) None

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) x=y=z=8

1 point

What will be the output?

```
#include<stdio.h>
int main()
{
int x;
  x= 9<5+3 && 7;
printf("%d", x);
return 0;
}
```

a. 0

b. 1

c. 7

d. Compilation error

No, the answer is incorrect.

Score: 0

Accepted Answers:

a. 0

Week 4 : Assignment

Week 4 : Assignment 4

The control/conditional statements used in C is/are

a) if-else statements

b) switch statements

c) Both (a) and (b)

d) None of these

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Both (a) and (b)

1 point

What is the output of the following C code?

```
#include <stdio.h>
int main()
  {
   int x = 1;
   if (x == 0)
     if (x >= 0)
       printf("x=0\n");
    else
      printf("x=1\n");
   return 0;
  }
```

a) x=1

b) x=0

c) Depends on compiler

d) No output

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) No output

1 point

Compute the printed value of i of the C program given below

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

a) 2

b) 3

c) 4

d) Compiler error

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) 2

1 point

If multiple conditions are used in a single if statement then the testing of those conditions are done

a) From Right to Left

b) From Left to right

c) Randomly

d) None

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) From Left to right

1 point

What is the purpose of the given program? n is the input number given by the user.

```
#include <stdio.h>
int main()
  {
   int n, x = 0, y;
   printf("Enter an integer: ");
   scanf("%d", &n);
   while (n != 0)
     {
      y = n % 10;
      x = x - y;
      n = n/10;
     }
   printf("Output is = %d", x);
   return 0;
}
```

a) Sum of the digits of a number

b) Negative sum of the digits of a number

c) Reverse of a number

d) The same number is printed

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) Negative sum of the digits of a number

1 point

while(1) is used in a program to create

a) False statement

b) Infinite loop

c) Terminating the loop

d) Never executed loop

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) Infinite loop

1 point

What will be the value of a, b, c after the execution of the followings

```
int a=5, b=7, c=111;
c /= ++a * b--;
```

a) a=5, b=6, c=2;

b) a=6, b=7, c=1;

c) a=6, b=6, c=2;

d) a=5, b=7, c=1;

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) a=6, b=6, c=2;

1 point

What will be printed when the following C code is executed?

```
#include<stdio.h>
int main()
{
   if('A'<'a')
   printf("NPTEL");
   else
   printf("PROGRAMMING");
   return 0;
}
```

a) NPTEL

b) PROGRAMMING

c) No output

d) Compilation error as A and a are not declared as character variable

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) NPTEL

1 point

switch statement in C, do not accept which data types from the list below:

a) int

b) char

c) long

d) enum

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) long

1 point

What will be the output of the following program?

```
#include <stdio.h>
int main()
{
  int x = 1;
  switch (x)
  {
     case 1: printf("Choice is 1 \n");
     default: printf("Choice other than 1 \n");
  }
  return 0;
}
```

a) Choice is 1

b) Choice other than 1

c) Both (a) and (b)

d) Syntax error

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Both (a) and (b)

Week 5 : Assignment

Week 5 : Assignment 5

In C language three way transfer of control is possible using

a) Ternary operator

b) Unary operator

c) Logical operator

d) None of the above are true

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) Ternary operator

1 point

The loop in which the statements within the loop are executed at least once is called

a) for

b) do-while

c) while

d) goto

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) do-while

1 point

What will be the output?

```
#include <stdio.h>
int main()
{
   float k = 0;
   for (k = 0.5; k < 3; k++)
      printf("I love C\n");
   return 0;
}
```

a) Error

b) I love C -- will be printed 6 times

c) I love C -- will be printed 3 times

d) I love C –will be printed 5 times

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) I love C -- will be printed 3 times

1 point

What is the output of the following code?

```
#include <stdio.h>
int main()
{
  int i=1;
  do
  {
   printf("while vs do-while\n");
  }while(i==1);
  printf("Out of loop");
   return 0;
}
```

a) ‘while vs do-while’ once

b) ‘Out of loop’ infinite times

c) Both ‘while vs do-while’ and ‘Out of loop’ once

d) ‘while vs do-while’ infinite times

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) ‘while vs do-while’ infinite times

1 point

Find the output of the following C program

```
#include <stdio.h>
int main()
{
int i = 0;
    if(i==0)
    {
   i=i+1;
   break;
   }
   printf("%d", i);
   return 0;
}
```

a) 0

b) 1

c) No output

d) Compiler error

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) Compiler error

1 point

What is the output of the following C program?

```
#include <stdio.h>
int main()
{
   int a = 0, i, b;
   for (i = 0; i <= 2; i+= 0.5)
   {
      a++;
      continue;
   }
   printf("%d", a);
   return 0;
}
```

a) 5

b) 4

c) 1

d) No output

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) No output

1 point

What will be the output?

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

a) Syntax error

b) 0 1 2 3 4 5 6 7 8 9 10

c) 1 2 3 4 5 6 7 8 9 10

d) 0 1 2 3 4 5 6 7 8 9

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 1 2 3 4 5 6 7 8 9 10

1 point

What is the output of the following C code?

```
#include <stdio.h>
int main()
{
   int a = 1;
   if (a--)
      printf("True\n");
   if (++a)
      printf("False\n");
   return 0;
}
```

a) True

b) False

c) Both ‘True’ and ‘False’

d) Compilation error

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Both ‘True’ and ‘False’

1 point

What is the output of the below C program?

```
#include <stdio.h>
int main()
{
   short int k=1, j=1;
   while (k <= 4 || j <= 3)
   {
                k=k+2;
                j+=1;
   }
printf("%d, %d", k,j);
return 0;
}
```

a) 5,4

b) 7,4

c) 5,6

d) 6,4

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) 7,4

1 point

What will be the value of ‘i’ after the execution of the program below

```
#include <stdio.h>
int main()
{
int i=1, j;
  for(j=0; j<=10; j+=i)
  {
       i=i+j;
  }
return 0;
}
```

a) 10

b) 11

c) 12

d) 13

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) 13

Week 6 : Assignment

Week 6 : Assignment 6

Which of the statements is correct?

a) An array contains more than one element

b) All elements of array have to be of same data type

c) The size of array has to be declared upfront

d) All of the above

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) All of the above

1 point

An integer array of size 15 is declared in a C program. The memory location of the first byte of the array is 2000. What will be the location of the 13th element of the array? Assume int data type takes 2 bytes of memory.

a) 2013

b) 2024

c) 2026

d) 2030

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) 2024

1 point

Which statement is correct?

a) An index or subscript in array is a positive integer

b) An index or subscript in array is a positive or negative integer

c) An index or subscript in array is a real number

d) None

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) An index or subscript in array is a positive integer

1 point

What will happen if in a C program you assign a value to an array element whose index exceeds the size of array?

a) The element will be set to 0

b) The compiler will not give any error but program may crash if some important data gets overwritten.

c) The compiler would report an error.

d) The array size would appropriately grow.

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) The compiler will not give any error but program may crash if some important data gets overwritten.

1 point

Which of the following statements is correct for the following code snippet?

```
int num[7];
```

num[7]=8;

a) In the first statement 7 specifies a particular element, whereas in the second statement it specifies a type;

b) In the first statement 7 specifies a particular element, whereas in the second statement it specifies the array size.

c) In the first statement 7 specifies the array size, whereas in the second statement it specifies a particular element of array.

d) In both the statement 7 specifies array size.

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) In the first statement 7 specifies the array size, whereas in the second statement it specifies a particular element of array.

1 point

What will be output of the following program?

```
int main()
{
   int i;
   int arr[3] = {3};
   for (i = 0; i < 3; i++)
      printf("%d ", arr[i]);
   return 0;
}
```

a) 3 followed by garbage values

b) 3 0 0

c) 3 1 1

d) Syntax error

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) 3 0 0

1 point

What will be the output?

```
#include <stdio.h>
int main()
{
   int arr[]={1,2,3,4,5,6};
   int i,j,k;
   j=++arr[2];
   k=arr[1]++;
   i=arr[j++];
   printf("i=%d, j=%d, k=%d", i, j, k);
return 0;
}
```

a) i=5, j=5, k=2

b) i=6, j=5, k=3

c) i=6, j=4, k=2

d) i=5, j=4, k=2

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) i=5, j=5, k=2

1 point

What will be the output after execution of the program?

```
#include <stdio.h>
int main()
{
   int i, a[4]={3,1,2,4}, result;
   result=a[0];
   for(i=1;i<4;i++)
   {
            if(result>a[i])
            continue;
            result=a[i];
   }
   printf("%d", result);
  return 0;
}
```

a) 1

b) 2

c) 3

d) 4

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) 4

What will be the output?

```
#include<stdio.h>
int main()
{
int n = 2;
int sum = 5;
switch(n)
{
        case 2: sum = sum-3;
        case 3: sum*=4;
        break;
        default:
                sum =0;

}
printf("%d", sum);
   return 0;
}
```

Hint

Yes, the answer is correct.

Score: 1

Accepted Answers:

(Type: Numeric) 8

1 point

1 point

Find the output of the following C program

```
#include<stdio.h>
int main()
{
        int a;
        int arr[5] = {1, 2, 3, 4, 5};
        arr[1] = ++arr[1];
        a = arr[1]++;
        arr[1] = arr[a++];
        printf("%d, %d", a, arr[1]);
        return 0;
}
```

a) 5,4

b) 5,5

c) 4,4

d) 3,4

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 4,4

Week 7 : Assignment

Week 7 : Assignment 7

Which of the following statements are correct?

1) A string is a collection of characters terminated by '\0'.

2) The format specifier %s is used to print a string.

3) The length of the string can be obtained by using the function strlen().

4) The pointer cannot work on string

a) 1,2

b) 1,2,3

c) 2,4

d) 1,3

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) 1,2,3

1 point

The correct method of initializing a 2D array is

int abc[2][2] = {1, 2, 3 ,4}

int abc[ ][ ] = {1, 2, 3 ,4}

int abc[2][ ] = {1, 2, 3 ,4}

All of the above

Yes, the answer is correct.

Score: 1

Accepted Answers:

int abc[2][2] = {1, 2, 3 ,4}

1 point

Array passed as an argument to a function is interpreted as

a) Address of all the elements in an array

b) Value of the first element of the array

c) Address of the first element of the array

d) Number of element of the array

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Address of the first element of the array

What will be the output?

```
#include <stdio.h>
int main()
{
int disp[3][4] = {{5, 6, 8, 2}, {4, 5, 3, 7}, {1,10,13,15}};
printf("%d\n", disp[2][1]);
return 0;
}
```

Hint

Yes, the answer is correct.

Score: 1

Accepted Answers:

(Type: Numeric) 10

1 point

1 point

Find the output of the following C program.

```
#include <stdio.h>
int main()
{
char a[10][8] = {"hi", "hello", "fellows"};
printf("%s", a[2]);
return 0;
}
```

a) fellows

b) h

c) fello

d) Compiler error

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) fellows

1 point

What will be the output?

```
#include <stdio.h>
int main()
{
char str1[] = "Week-7-Assignment";
char str2[] = {'W', 'e', 'e', 'k', '-', '7', '-', 'A', 's','s','i','g','n','m','e','n','t'};
int n1 = sizeof(str1)/sizeof(str1[0]);
int n2 = sizeof(str2)/sizeof(str2[0]);
printf("n1 = %d, n2 = %d", n1, n2);
return 0;
}
```

a) n1=18, n2=17

b) n1=18, n2=18

c) n1=17, n1=17

d) n1=17, n2=18

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) n1=18, n2=17

1 point

Consider the following C program segment:

```
#include<stdio.h>
#include<string.h>
int main()
{
char p[20];
char s[] = "string";
int length = strlen(s);
int i;
for (i = 0; i < length; i++)
    p[i] = s[length - i];
printf("%s", p);
return 0;
}
```

The output would be-

a) gnirts

b) gnirt

c) string

d) Nothing is printed

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) Nothing is printed

1 point

If the starting address of a floating point array Arr[10][10] is 2000,

what would be the memory address of the element Arr[5][6]? (Considering float takes 4 bytes of memory)

a) 2268

b) 2120

c) 2224

d) 2144

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 2224

1 point

In C, the placement of elements of a two dimensional array is

a) Row wise

b) Column wise

c) Diagonal wise

d) Bottom to top wise

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) Row wise

What will be the value of ‘i’ after the execution of the C code fragment given below?

```
static char str1[] = "dills";
static char str2[20];
static char str3[] = "daffo";
int i;
i = strcmp(strcat(str3, strcpy(str2, str1)), "daffodills");
```

Hint

Yes, the answer is correct.

Score: 1

Accepted Answers:

(Type: Numeric) 0

Week 8 : Assignment

Week 8 : Assignment 8

A function prototype is used for

a) Declaring the function logic

b) Calling the function from the main body

c) Telling the compiler, the kind of arguments used in the function

d) Telling the user for proper use of syntax while calling the function

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Telling the compiler, the kind of arguments used in the function

1 point

What is the output of the following C program?

```
#include <stdio.h>
void foo(), f();
int main()
{
  f();
return 0;
}
void foo()
{
   printf("2 ");
}
void f()
{
   printf("1 ");
   foo();
}
```

a) Compiler error as foo() is not declared in main

b) 1 2

c) 2 1

d) Compile time error due to declaration of functions inside main

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) 1 2

1 point

How many times ‘Hi’ will be printed in the program given below

```
#include<stdio.h>
int i;
int fun();

int main()
{
   while(i)
   {
      fun();
      main();
   }
   printf("Hello\n");
   return 0;
}
int fun()
{
   printf("Hi");
}
```

a) Only once

b) Zero times

c) Infinite times

d) Compilation error

Yes, the answer is correct.

Score: 1

Accepted Answers:

b) Zero times

1 point

What is the output of the C code given below

```
#include <stdio.h>
float func(float age[]);

int main()
{
   float result, age[] = { 23.4, 55, 22.6, 3, 40.5, 18 };
   result = func(age);
   printf("%0.2f", result);
   return 0;
}

float func(float age[])
{
   int i;
   float result, sum = 0.0;
   for (i = 0; i < 6; ++i) {
      sum += age[i];
   }
   result = (sum / 6);
   return result;
}
```

a) 27.08

b) 27.083334

c) Compiler error as result is declared twice

d) Error: Invalid prototype declaration

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) 27.08

1 point

Which statement is correct about Passing by value parameters?

a) It cannot change the actual parameter value

b) It can change the actual parameter value

c) Parameters is always in read-write mode

d) None of them

Yes, the answer is correct.

Score: 1

Accepted Answers:

a) It cannot change the actual parameter value

1 point

What will be the output?

```
int main()
{
    {
        int a = 70;
     }
     {
        printf("%d", a);
     }
  return 0;
}
```

a) 70

b) Garbage value

c) Compilation error

d) None

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) Compilation error

1 point

How many times Hello world will be printed?

```
#include<stdio.h>
int main()
{
   printf("Hello world\n");
   main();
   return 0;
}
```

a) Infinite times

b) 32767

c) 65535

d) Till stack overflows

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) Till stack overflows

1 point

What will be the output?

```
#include<stdio.h>
void func(int n, int sum)
{
int k = 0, j = 0;
if (n == 0) return;
        k = n % 10;
j = n / 10;
sum = sum + k;
func (j, sum);
printf ("%d, ", k);
}

int main ()
{
int a = 2048, sum = 0;
func (a, sum);
printf ("%d ", sum);
return 0;
}
```

a) 8 ,4, 0, 2, 14

b) 8, 4, 0, 2, 0

c) 2, 0, 4, 8. 14

d) 2, 0, 4, 8, 0

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) 2, 0, 4, 8, 0

1 point

Consider the function

```
find(int x, int y)
{
        return((x<y) ? 0 : (x-y));
}
```

Let a and b be two non-negative integers. The call find(a, find(a, b)) can be used to find the

a) Maximum of a, b

b) Positive difference between a and b

c) Sum of a and b

d) Minimum of a and b

Yes, the answer is correct.

Score: 1

Accepted Answers:

d) Minimum of a and b

1 point

Consider the function

```
int fun(x: integer)
{
if x > 100 then fun = x – 10;
else
fun = fun(fun(x + 11));
}
```

For the input x = 95, the function will return

a) 89

b) 90

c) 91

d) 92

Yes, the answer is correct.

Score: 1

Accepted Answers:

c) 91

Week 9 : Assignment

Week 9 : Assignment 9

Which of the following are C pre-processor directive?

a) #ifdef

b) #define

c) #endif

d) all of the mentioned

1 point

What is the correct order of insertion sort (in ascending order) of the array arr[5]={8 3 5 9 4}?

a) {3 8 5 9 4} --> {3 5 8 9 4} --> {3 4 5 8 9}

b) {3 8 5 9 4} --> {3 5 8 9 4} --> {3 5 8 4 9} --> {3 5 4 8 9} --> {3 4 5 8 9}

c) {3 8 5 9 4} --> {3 4 8 5 9} --> {3 4 5 8 9} --> {3 4 5 8 9} --> {3 4 5 8 9}

d) {8 3 5 4 9} --> {8 3 4 5 9} --> {3 4 5 8 9}

1 point

Given an array arr = {12, 34, 47, 62, 85, 92, 95, 99,105} and key = 34; what are the mid values (corresponding array elements) generated in the first and second iterations of binary search?

a) 85 and 12

b) 85 and 34

c) 62 and 34

d) 62 and 47

1 point

Binary Search can be categorized into which of the following?

a) Brute Force technique

b) Divide and conquer

c) Greedy algorithm

d) Dynamic programming

1 point

Select the code snippet which performs unordered linear search iteratively?

```
int unorderedLinearSearch(int arr[], int size, int data)
{
   int index;
   for(int i = 0; i < size; i++)
   {
      if(arr[i] == data)
      {
         index = i;
         break;
      }
   }
   return index;
}
```

```
int unorderedLinearSearch(int arr[], int size, int data)
{
   int index;
   for(int i = 0; i < size; i++)
   {
      if(arr[i] == data)
      {
         break;
      }
   }
   return index;
}
```

```
int unorderedLinearSearch(int arr[], int size, int data)
{
   int index;
   for(int i = 0; i <= size; i++)
   {
      if(arr[i] == data)
      {
         index = i;
         continue;
      }
   }
   return index;
}
```

d) None of the above

1 point

Consider the array A[]= {5,4,9,1,3} apply the insertion sort to sort the array . Consider the cost associated with each sort is 25 rupees, what is the total cost of the insertion sort for sorting the entire array?

a) 25

b) 50

c) 75

d) 100

1 point

What will be the output of the following C code?

```
#include <stdio.h>
#if A == 1
   #define B 0
#else
   #define B 1
#endif
int main()
{
   printf("%d", B);
   return 0;
}
```

a) 0

b) 1

c) 01

d) None of the above

1 point

What will be the output?

```
#include <stdio.h>
#define a 10
int main()
{
  printf("%d ",a);
  int a=50;
  printf("%d ",a);
  return 0;
}
```

a) 10 10

b) 10 50

c) 50 50

d) Compilation error

1 point

What is the worst case complexity of selection sort?

O(nlogn)

O(logn)

O(n)

$O(n^2)$

1 point

If the given input array is sorted or nearly sorted, which of the following algorithm gives the best performance?

a) Insertion sort

b) Selection sort

c) Bubble sort

d) Quick sort

Week 10 : Assignment

Week 10 : Assignment 10

How can you improve the best-case efficiency in bubble sort? (The input is already sorted)

a)
```
boolean swapped = false;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        swapped = true;
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                        swapped = false;
                }
        }
}
```

b)
```
boolean swapped = true;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        swapped = false;
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                }
        }
}
```

c)
```
boolean swapped = true;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        swapped = false;
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                        swapped = true;
                }
        }
}
```

d)
```
boolean swapped = true;
for(int j=arr.length-1; j>=0 && swapped; j--)
{
        for(int k=0; k<j; k++)
        {
                if(arr[k] > arr[k+1])
                {
                        int temp = arr[k];
                        arr[k] = arr[k+1];
                        arr[k+1] = temp;
                        swapped = true;
                }
        }
}
```

1 point

Bisection method is used to find

a) Derivative of a function at a given point

b) Numerical integration of a function within a range

c) Root of a function

d) None of the above

1 point

In ……., search starts at the beginning of the list and checks every element in the list.

a) Linear search

b) Binary search

c) Hash search

d) Binary tree search

1 point

What is the output of the following program?

```
# include <stdio.h>
void func(int x)
{
   x = 40;
}

int main()
{
  int y = 30;
  func(y);
  printf("%d", y);
  return 0;
}
```

a) 40

b) 30

c) Compilation error

d) Runtime error

Simple way programs

Check if number is positive or negative

```
#include <stdio.h>
 int main() {
      int n;
      scanf("%d", &n);
      printf(n >= 0 ? "Positive\n" : "Negative\n");
      return 0;
 }
```

Check if a number is even or odd

```
#include <stdio.h>
 int main() {
      int n;
      scanf("%d", &n);
      printf(n % 2 == 0 ? "Even\n" : "Odd\n");
      return 0;
 }
```

Find maximum of two numbers

```
#include <stdio.h>
 int main() {
      int a, b;
      scanf("%d %d", &a, &b);
      printf("%d\n", (a > b) ? a : b);
      return 0;
 }
```

Check if character is vowel

```
#include <stdio.h>
 int main() {
      char c;
      scanf(" %c", &c);
      if (c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||
         c=='A'||c=='E'||c=='I'||c=='O'||c=='U')
      printf("Vowel\n");
      else
      printf("Not Vowel\n");
      return 0;
 }
```

Print ASCII value of character

```
#include <stdio.h>
 int main() {
      char c;
      scanf(" %c", &c);
      printf("%d\n", c);
      return 0;
 }
```

Swap two numbers (using temp variable)

```
#include <stdio.h>
 int main() {
      int a, b, t;
      scanf("%d %d", &a, &b);
      t = a; a = b; b = t;
      printf("%d %d\n", a, b);
      return 0;
 }
```

Calculate square of a number

```
#include <stdio.h>
 int main() {
      int n;
      scanf("%d", &n);
      printf("%d\n", n * n);
      return 0;
 }
```

Print first character of a string

```
#include <stdio.h>
 int main() {
      char str[100];
      scanf("%s", str);
      printf("%c\n", str[0]);
      return 0;
 }
```

Check if number is divisible by 5

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

```
      int n;
      scanf("%d", &n);
      printf(n % 5 == 0 ? "Yes\n" : "No\n");
      return 0;
 }
```

Multiply two numbers

```
#include <stdio.h>
 int main() {
      int a, b;
      scanf("%d %d", &a, &b);
      printf("%d\n", a * b);
      return 0;
 }
```

Tab 35

```c
#include <stdio.h>          //positive or negative
int main() {
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);                  // Input a number
  // Check if the number is positive or negative
   if (num >= 0) {
      printf("The number is positive or zero.\n");
   } else {
      printf("The number is negative.\n");
   }
   return 0;
}
```

```c
int main() {                  //vowel or consonant
   char ch;
   printf("Enter a character: ");
   scanf("%c", &ch);                  // Input a character
   // Check if the character is a vowel or consonant
   if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
      ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
      printf("%c is a vowel.\n", ch);
   } else {
      printf("%c is a consonant.\n", ch);
   }
   return 0;
}
```

```c
printf("Enter a number: ")              // even or odd
      scanf("%d", &num);
   if (num % 2 == 0) {
      printf("The number %d is even.\n", num);
   } else {
      printf("The number %d is odd.\n", num)
}
```

output:

```c
int main() {                    // Leap Year Checker
  int year;
    printf("Enter a year: ");      // Input a year
    scanf("%d", &year);      // Check for leap year
    if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0))) {
        printf("%d is a leap year.\n", year);
    } else {
        printf("%d is not a leap year.\n", year);
    }
    return 0;
}
```

```c
int main() {                    //Grade Checker (Using if-else ladder)
    int marks;

    // Input marks
    printf("Enter marks: ");
    scanf("%d", &marks);
    if (marks >= 90) {                // Check the grade using an if-else
ladder
        printf("Grade: A\n");
    } else if (marks >= 75) {
        printf("Grade: B\n");
    } else if (marks >= 50) {
        printf("Grade: C\n");
    } else if (marks >= 35) {
        printf("Grade: D\n");
    } else {
        printf("Grade: F\n");
    }
    return 0;
}
```

```c
    if (age >= 18) {
```

```
    printf("You are eligible to vote.\n");     // Check voting eligibility
  } else {
    printf("You are not eligible to vote.\n");
  }
```

```
int main() {                    //Simple Bank Account Balance Check
  double balance, withdraw;
  printf("Enter your account balance: ");     scanf("%lf", &balance);
  printf("Enter the withdrawal amount: ");   scanf("%lf", &withdraw);
  if (withdraw <= balance) {          // Check if withdrawal is possible
    printf("Withdrawal successful! Your remaining balance is %.2lf\n", balance - withdraw);
  } else {
    printf("Insufficient funds! You cannot withdraw %.2lf.\n", withdraw);
  }
```

```
int main() {                    //Divisibility Checker (Complex condition)
  int num;  // Input a number
  printf("Enter a number: ");
  scanf("%d", &num);  //Check if the number is divisible by both 3 and 5
  if (num % 3 == 0 && num % 5 == 0) {
    printf("The number %d is divisible by both 3 and 5.\n", num);
  } else {
    printf("The number %d is not divisible by both 3 and 5.\n", num);
  }
  return 0;
}
```

```
int main() {                                  // Largest of two numbers
  int num1, num2;
  printf("Enter two numbers: ");
  scanf("%d %d", &num1, &num2);
                                  // Check which number is largest
  if (num1 > num2) {
    printf("The largest number is %d.\n", num1);
  } else if (num2 > num1) {
    printf("The largest number is %d.\n", num2);
```

```
    } else {
        printf("Both numbers are equal.\n");
    }
 return 0;
}
```

```
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
  // Check if the number is even or odd using the conditional
operator
(num % 2 == 0) ? printf("The number %d is even.\n", num) : printf("The number %d is odd.\n", num);
    return 0;
}
```

```
printf("Enter a number: ");
    scanf("%d", &num);
  // Check if the number is positive, negative, or zero
    if (num > 0) {
        printf("The number %d is positive.\n", num);
    } else if (num < 0) {
        printf("The number %d is negative.\n", num);
    } else {
        printf("The number is zero.\n");
    }
```

```
int main() {
    int a = 1, b = 0, c = -1;
    if (a > b && b > c || c == 0) {
        printf("Condition is TRUE\n");
    } else {
        printf("Condition is FALSE\n");
    }
```

```
int main() {
    int x = 5, y = 10;
```

```
  if ((x = y) > 5) { // Assignment within the condition
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

```
int main() {
   int a = 5, b = 3;
   if (a & b == 1) {                  // Bitwise AND (&) with equality (==)
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

```
int main() {
   int x = 4, y = 2, z = 0;

  if (x | y && z) {                          // Logical AND and bitwise OR
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

Output:

Condition is FALSE

```
int main() {     // confusing ELSE condition
   int a = 5, b = 10;
   if (a > 0)
        if (b < 5)
          printf("Inner condition is TRUE\n");
  else  // Misleading else
         printf("Inner condition is FALSE\n");

   return 0;
}
```

Output:

Inner condition is FALSE

```
int main() {
   int x = 0, y = 10;

   if ( !(x && y || !y)) {              // Negations combined with logical
operators
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }  return 0;
}
```

Output:

Condition is TRUE

```
#include<stdio.h>
int main() {
   int i = 0;
   if (i++ == 0 && i++ == 1) {              // Post-increment during evaluation
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
   printf("Final value of i: %d\n", i);
   return 0;
}
```

output:

Condition is TRUE

Final value of i: 2

```
int main() {
   int a = 5, b = 10;
   if ((a > b ? a : b) < (b > a ? b : a)) {   // Nested ternary operators
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
```

```
    }
    return 0;
}
```

Output:

Condition is FALSE

```
 int main()
{ int i=0;  // int i=1;
int condition1 = (i++ == 0);     // Assume i is initialized as   i=0  and another case i=1
int condition2 = (i++ == 1);
if ( condition1 && condition2) {
    printf("Condition is TRUE\n");
} else {
    printf("Condition is FALSE\n");
}
printf("Final value of i : %d\n", i);
return 0;
}
```

Output:

```
 // when i=0
Condition is TRUE
 Final value of i : 2
 // when i=1
Condition is FALSE
Final value of i : 3
```

```
int main() {
 int  a=2, b=3 ;
int max1 = (a > b) ? a : a*a;
int max2 = (b > a) ? b+max1 : b ;
    printf("%d %d ",max1,max2);
if (max1 < max2) {
    printf("Condition is TRUE\n");
} else {
```

```
    printf("Condition is FALSE\n");
}
    printf("%d %d ",max1,max2);
}
```

Output:

4 7

Condition is TRUE

4 7

```
int main() {
    int x = 2147483647;             // Maximum value for a 32-bit signed integer

    if (x + 1 < x) {     // Overflow happens here
        printf("Condition is TRUE\n");
    } else {
        printf("Condition is FALSE\n");
    }
    return 0;
}
```

output :

Condition is True // for 32 bit systems

Condition is FALSE // for 64 bit systems

```
int main() {
    char ch = 255;                  // 255 exceeds the range of a signed char (-128 to 127)

    if (ch > 0) {
        printf("Condition is TRUE\n");
    } else {
        printf("Condition is FALSE\n");
    }
    return 0;
}
```

Output:

Condition is FALSE

Reason:

255 is outside the range of a signed `char`, so it wraps around.

On most systems using two's complement, this happens:

`char ch = 255; → ch becomes -1 ( via two's compliment )`

255 in binary: `11111111`

As a signed char, this is interpreted as `-1`

```
int main() {
   int a = 1, b = 2;
   if (a = b, a == 1) {          // Comma operator in the condition
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
printf( " %d ", a );
   return 0;
}
```

Output:

Condition is FALSE

2

```
int main() {
   unsigned int a = 1;
   int b = -1;

   if (a < b) {                              // Signed v/s unsigned comparison
      printf("Condition is TRUE\n");
   } else {
      printf("Condition is FALSE\n");
   }
```

```
    return 0;
}
```

```
int x = 2147483647;                    // Max value for 32-bit signed int

if (x + 1 < x) {
    printf("Condition is TRUE\n");
} else {
    printf("Condition is FALSE\n");
}
```

```
#include<stdio.h>
int main() {
   if ( !!(x > 5)) {
    printf("True\n");
   } else {
    printf("False\n");
      }
}
```

//Will this condition be true or false for x = 10?

True

```
//Mixed Operators*
int main( ) {
if (x > 5 && !x || x < 0) {
    printf("True\n");
} else {
    printf("False\n");
}
}
```

//What will be the output for x = 10 ?

False

```
// * Ternary Operator*
int main( ) {
    int x = 10;
     int y = 5;
   if ((x < y) ? x : y > 10) {
        printf("True\n");
     } else {
        printf("False\n");
     }
}
```

//Will this condition be true or false?

False

```
//    * Bitwise Operations
int x = 5;
if ((x & 1) && (x >> 1)) {
    printf("True\n");
} else {
    printf("False\n");
}
```

What's the output?

```
//* Pointer Comparison*

int arr[5] = {1, 2, 3, 4, 5};
int *p = arr;
if (p + 1 > p) {                    // addresses are compared
    printf("condition becomes True \n");
} else {
    printf("Condition becomes False\n");
```

```
}
```

//Is this condition true or false?

Output:

Condition becomes True

//*3. **Enum Comparison***

```
enum Color { RED, GREEN, BLUE };   // here RED- 0 GREEN - 1  BLUE - 2
enum Color c = GREEN;                    // now c  will get 1
if (c > RED && c < BLUE) {
    printf("True\n");
} else {
    printf("False\n");
}
```

What's the output?

True

//***4. Floating-Point Comparison***

```
int main( ) {
  float x = 0.1 + 0.2;
if (x == 0.3) {                          //   ( floating point numbers are stored approximately )
    printf("True\n");
} else {
    printf("False\n");
}
return 0;
}
```

//Is this condition True or false?

False

//*5. **Array Indexing***

```
int arr[5] = {1, 2, 3, 4};
if (arr[2] > arr[4]) {
```

```
    printf("True\n");
} else {
    printf("False\n");
}
// What's the output?
```

Output :

True

```
int main() {
    int i;
    if(i=0,2,3) {         // This is the tricky part!
        printf(" Do you know NPTEL\n");
    } else {
        printf("Programming on C ");
    }
    printf("%d\n", i);
    return 0;
}
```

reason:

This uses the **comma operator** (,). The comma operator evaluates its operands from left to right and the **result of the entire expression is the value of the rightmost operand which is 3.**

```
#include <stdio.h>

int main() {
    int x = 0;
    if (x++) {
        printf("true\n");
    } else if (x == 1) {
        printf("false\n");
    }
    return 0;
}
```

Output:

false

```
#include <stdio.h>

int main() {
   int a = 1, b = 2, c = 3;
   if (c > b > a) {     // This condition will be evaluated due to operator associativity
      printf("true");
   } else {
      printf("false");
   }
   return 0;
}
```

Output:

false

```
int main() {
   int a = 1, b = -1, c = 0, d;
   // Evaluate the expression for 'd' using logical operators and
pre/post-increment/decrement.
   // The '&&' (logical AND) and '||' (logical OR) operators short-circuit.
   // '++a' increments 'a' before its value is used in the expression.
   // '++b' increments 'b' before its value is used in the expression.
   // 'c--' uses the current value of 'c' in the expression, then decrements 'c'.
   d = ++a && ++b || c--;

   // Check the value of 'd'
   if (d) {                                  // If 'd' is non-zero (true)
      printf("Kolkata \n");
   } else if (c) {                           // If 'd' is zero (false), then check the value of 'c'
      printf("Delhi \n");
   } else {                                  // If both 'd' and 'c' are zero (false)
      printf("Bangalore \n");
   }
```

```
    return 0;
}
```

Output:

Delhi

```
int main() {
    int a = 1;
    if (a--) {
        printf("True\n");       // Since 1 is non-zero (true), this line executes
    }

    if (++a) {
        printf("False\n");
    }

    return 0; }
```

Output:

True

False

```
#include <stdio.h>

int main()
{
    if ((0 && 1) || (1 && -1))
        printf("Condition is true.");
    else
        printf("Condition is false.");
    return 0;
}
```

Output:

Condition is true

—----------

—----------

—----------

—----------

—----------

Little guess programs

—----------

if (**expression**)

statement ;

Examples:

```
if ( 3 + 2 % 5 )
    printf ( "yes this will work" ) ;
if ( a = 10 )
    printf ( "why not this one?" ) ;
if ( -5 )
    printf ( "Hush, even negative also works" ) ;
```

—----------

```
main( ) {
int i ;
printf ( "Enter value of i " ) ;
scanf ( "%d", &i ) ;
if ( i = 5 )
   printf ( "You entered 5" ) ;
else
   printf ( "You entered something other than 5" ) ;
}
```

Output:

Enter value of i 100

You entered 5

What will happen if we write a semicolon at the end of if.

```
if ( i == 5 ) ;
    printf ( "You entered 5" ) ;
return 0; }
```

Output:

Enter Value of i 10000

You entered 5

—----------

```
main( ) {
 float a = 12.25, b = 12.52 ;
 if ( a = b )
 printf ( "a and b are equal" ) ;
}
```

Output:

a and b are equal

—----------

```
#include<stdio.h>

   void main( ) {
 int x = 10, y = 15 ;
 if ( x  = y % 3 )
     printf ( " Don't you think it is Com.Err..." ) ;
}
```

Output:

Nothing will be printed

—----------

```
void main( ) {
char a = 'A' ;
 printf ( "%c  %d " , ( a <= 'a' ? a : '!' ) , a ) ;
}
```

Output:

A 65

—----------

What values of i, j variables could hold?

```
void main( )
{
 int j = 20, k = 22 ;
 if ( k >= j )
 {
      {
          k = j ;
          j = k ;
     }
 }
 printf("%d %d",k,j);
}
```

Output:

20 20

—----------

```
void main( )
{
 int i = -1, j = 1, k ,l ;
 k = !i && j ;
 l = !i || j ;
 printf ( "%d %d", k, l ) ;
}
```

Output:

0 1

```
#include<stdio.h>

void main( )
{
 int i = -3, j = 3 ;
 if ( !i + !j * 1 )
 printf ( "\nMIC" ) ;
 else
 printf ( "\nCollege" ) ;
}
```

Output:

College

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

—----------

quiz, arrays

Fill in the blanks

1. —---------- Keeps track of Intermediate values
2. Often we need to keep track of a collection of values. What is the possible one which helps for this case.
   a) Variable b) constant c) tokens d) array

3. To keep track of the latest value by using a
   a) Integer b) array c) variable d) char
4. **Count**
   Counting the objects which passes through the iteration
5. **sum**
   Summinng the each passed value
6. **Filtering**
   Filter the ones which we need and leave out the remaining. This is happening inside an iteration.

Case 1 : Counting Even Numbers in an Array (Filtering + Counting)

This program filters out the odd numbers and counts only the even numbers in an array. The filtering condition is `arr[i] % 2 == 0`.#include <stdio.h>

Case 2: Finding the First Negative Number (Filtering + Early Exit)

This program filters for the first negative number and stops the iteration once found. The filtering condition is `numbers[i] < 0`.#include <stdio.h>

Case 3: finding the number of students whose marks are greater than 35

Case 4: Finding the number(count) of students whose marks are greater than average marks.

Case 5: Finding the Modal value, means how many students have the same marks (2D array)

Case 6:

Examples for filtering:

If I have an array of 66 students, where I need to find out the maximum marks of a student. [keep track of the maximum]

To find the maximum marks gained student, I need to look until the end of the list and Stop if all are visited.

Program :

```
// assume first student's marks as maximum
  max = marks[0];

  // traverse the array to find maximum
   for(i = 1; i < 66; i++)
      {
          if(marks[i] > max){
```

```
                    max = marks[i]; //update max if current marks are higher
                }
            }
        printf(" The maximum marks of a student is ", max) ;
```

C Programs based on the Filtering Technique

Example 1: Finding the Maximum Value in an Array (Refinement of Existing Example)

This program finds the largest element in a given array of integers. The filtering condition is `marks[i] > max`.#include <stdio.h>

```
int main() {

    int marks[] = {55, 89, 72, 95, 61, 88};

    int n = 6; // Number of students/marks

    int max;

    int i;

    // 1. Initialization: Assume the first element is the maximum

    max = marks[0];

    // 2. Iteration and Filtering (Comparison)

    for (i = 1; i < n; i++) {

        // Filtering condition: Is the current element greater than the recorded max?

        if (marks[i] > max) {

            // Update (Keep track of the new maximum)

            max = marks[i];

        }
```

```
    }

    printf("The maximum marks of a student is: %d\n", max);

    return 0;

}
```

Example 2: Counting Even Numbers in an Array (Filtering + Counting)

This program filters out the odd numbers and counts only the even numbers in an array. The filtering condition is `arr[i] % 2 == 0`.#include <stdio.h>

```
int main() {

    int arr[] = {12, 7, 34, 19, 50, 11, 8};

    int n = 7;

    int count = 0; // Initialize counter

    int i;

    printf("Filtering (Counting) Even Numbers:\n");

    // Iteration

    for (i = 0; i < n; i++) {

        // Filtering condition: Check if the number is even

        if (arr[i] % 2 == 0) {

            // Keep track of the count (Counting the objects that pass the filter)

            count++;

            printf("Found even number: %d\n", arr[i]);

        }

    }

    printf("\nTotal number of even elements found: %d\n", count);

    return 0;
```

```
}
```

Example 3: Finding the First Negative Number (Filtering + Early Exit)

This program filters for the first negative number and stops the iteration once found. The filtering condition is `numbers[i] < 0`.#include <stdio.h>

```
#include <stdbool.h> // for 'bool' type

int main() {

    int numbers[] = { 10, 25, 5, -15, 30, -50 };

    int n = 6;

    int first_negative = 0;

    bool found = false;

    int i;

    // Iteration

    for (i = 0; i < n; i++) {

        // Filtering condition: Check if the number is negative

        if (numbers[i] < 0) {

            // Keep track of the value

            first_negative = numbers[i];

            found = true;

            // Stop (Early Exit)

            break;

        }

    }

    if (found) {

        printf("The first negative number found is: %d\n", first_negative);
```

```
    } else {
        printf("No negative numbers found in the array.\n");
    }
    return 0;
}
```

More Case Scenarios for Array Computation (Filtering, Counting, Summing, Tracking)

Case 4: Summing All Positive Numbers (Filtering + Summation)

This program iterates through an array and calculates the sum of only the positive numbers, ignoring zero and negative numbers.

- **Core Action:** Summation
- **Filtering Condition:** `arr[i] > 0`

Case 5: Checking if an Element Exists (Filtering + Boolean Tracking + Early Exit)

This program checks if a specific target value (e.g., 42) is present in the array. It uses a boolean flag and stops immediately once the element is found.

- **Core Action:** Tracking/Early Exit
- **Filtering Condition:** `arr[i] == target_value`

Case 6: Counting Multiples of 3 (Filtering + Counting)

This program counts how many elements in the array are perfectly divisible by 3.

- **Core Action:** Counting
- **Filtering Condition:** `arr[i] % 3 == 0`

Case 7: Finding the Index of the Smallest Element (Tracking Value and Index)

This program finds the minimum value in the array and, importantly, keeps track of the index (position) where that minimum value was found.

- **Core Action:** Tracking (both value and index)
- **Filtering Condition:** `arr[i] < min_value`

Case 8: Separating Odd and Even Numbers into Different Arrays (Filtering + Collection)

This program filters the elements of one array and copies the odd numbers into one new array and the even numbers into another new array.

- **Core Action:** Filtering/Collection
- **Filtering Condition 1 (Even):** `arr[i] % 2 == 0`
- **Filtering Condition 2 (Odd):** `arr[i] % 2 != 0`

Case 9: Calculating the Average of Numbers within a Range (Filtering + Summation + Counting)

This program calculates the average of only those numbers in the array that fall within a specific range (e.g., between 50 and 100, inclusive). It requires both summing the filtered elements and counting them.

- **Core Action:** Summation and Counting
- **Filtering Condition:** `arr[i] >= 50 && arr[i] <= 100`

for loop_quiz

```
int main() {
    int n=5;

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

    return 0;
}
```

```
int main() {
   int i=1;
   for( printf( " Namasthe \n ") ;  i< 3 ;  printf("   IP chaltha %d \n",i++)  )
{
        if( i==1)
             printf(" kya chaltha hai?\n " );
        else
            {
              printf(" IP acha hai ?\n ");
             break;
            }
}
printf(" acha acha.... " );
   return 0;
}
```

```
void main( ) {
    int i;
for ( i = 10 ; i ; i -- )
printf ( "%d ", i ) ;
}
```

Output:

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

```
#include<stdio.h>
void main( ) {
    int i,j;
for ( i = 10, j=5 ; j<=6 ; j++,i-- )
   printf ( "%d %d \n", i,j ) ;
}
```

Output:

```
10 5
9 6
```

Damodar Kammili

while questions

```
int main() {
    int n, i = 1;
    scanf("%d", &n);
    while (i <= n)
            printf("%d ", i++);
    return 0;
}
```

Important Number Logic Programs in C

Important Number Logic Programs in C

1. Reverse a Number

Example:

```
Input → 1234
Output → 4321
```

```
#include <stdio.h>

int main()
{
    int num, rev = 0, rem;

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

    while(num != 0)
    {
        rem = num % 10;
        rev = rev * 10 + rem;
        num = num / 10;
    }

    printf("Reverse = %d", rev);

    return 0;
}
```

2. Palindrome Number

(Number same after reverse)

Example: 121, 1331

```
#include <stdio.h>

int main()
{
    int num, temp, rev = 0, rem;

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

    temp = num;

    while(num != 0)
    {
        rem = num % 10;
        rev = rev * 10 + rem;
        num = num / 10;
    }

    if(temp == rev)
        printf("Palindrome");
    else
        printf("Not Palindrome");

    return 0;
}
```

---

3. Armstrong Number

(Example: 153 → $1^3 + 5^3 + 3^3 = 153$)

```
#include <stdio.h>
#include <math.h>

int main()
{
    int num, temp, rem, sum = 0;

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

```
    temp = num;

    while(num != 0)
    {
        rem = num % 10;
        sum = sum + pow(rem,3);
        num = num / 10;
    }

    if(sum == temp)
        printf("Armstrong Number");
    else
        printf("Not Armstrong");

    return 0;
}
```

---

4. Sum of Digits

Example: 123 → 1+2+3 = 6

```
#include <stdio.h>

int main()
{
    int num, sum = 0, rem;

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

    while(num != 0)
    {
        rem = num % 10;
        sum += rem;
        num /= 10;
    }

    printf("Sum of digits = %d", sum);

    return 0;
}
```

---

5. Count Number of Digits

```
#include <stdio.h>

int main()
{
    int num, count = 0;

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

    while(num != 0)
    {
        num = num / 10;
        count++;
    }

    printf("Total digits = %d", count);

    return 0;
}
```

---

6. Strong Number

```
(145 → 1! + 4! + 5! = 145)

#include <stdio.h>

int main()
{
    int num, temp, rem, sum = 0, fact, i;

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

    temp = num;

    while(num != 0)
    {
        rem = num % 10;
        fact = 1;

        for(i=1; i<=rem; i++)
            fact *= i;

        sum += fact;
```

```
        num /= 10;
    }

    if(sum == temp)
        printf("Strong Number");
    else
        printf("Not Strong Number");

    return 0;
}
```

---

7. Prime Number

```
#include <stdio.h>

int main()
{
    int num, i, flag = 0;

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

    for(i=2; i<=num/2; i++)
    {
        if(num % i == 0)
        {
            flag = 1;
            break;
        }
    }

    if(num == 1)
        printf("Not Prime");
    else if(flag == 0)
        printf("Prime");
    else
        printf("Not Prime");

    return 0;
}
```

---

8. Fibonacci Series

```
#include <stdio.h>

int main()
{
    int n, a=0, b=1, c, i;

    printf("Enter number of terms: ");
    scanf("%d", &n);

    for(i=1; i<=n; i++)
    {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }

    return 0;
}
```

Placement Important Order

(for Solving the problems quickly step by step)

1. Sum of digits
2. Reverse number
3. Palindrome
4. Armstrong
5. Prime
6. Fibonacci
7. Strong number

Logic Pattern Behind Most Number Programs

You should remember:

👉 **Extract digit → `num % 10`**
👉 **Remove digit → `num / 10`**
👉 **Loop until → `num != 0`**