1(A). Design an algorithm for area and perimeter of square.

Algorithm:

Step 1: Start

2: Read the side (Length of one side of the square)

3: area = side * side

4: perimeter = 4 * side

5: print area and perimeter

6: stop

1(B). Discuss about the tokens of c programming.

A token in C can be defined as the smallest individual element of the C programming language that is meaningful to the compiler. It is the basic component of a C program.

Types of Tokens in C

The tokens of C language can be classified into six types based on the functions they are used to perform. The types of C tokens are as follows:

1.Keywords 2. Identifiers 3.Constants

6. Strings 5. Special Symbols 6. Operators

1(C). Differentiate between type casting and type conversion.

S.NO	TYPE CASTING	TYPE CONVERSION
1.	In type casting, a data type is converted into another data type by a programmer using casting operator.	Whereas in type conversion, a data type is converted into another data type by a compiler.
2.	Type casting can be applied to **compatible data types** as well as **incompatible data types**.	Whereas type conversion can only be applied to **compatible datatypes**.
3.	Type casting takes place during the program design by programmer.	Whereas type conversion is done at the compile time.
4.	Type casting is also called narrowing conversion because in this, the destination data type may be smaller than the source data type.	Whereas type conversion is also called widening conversion because in this, the destination data type can not be smaller than the source data type.

1(D). Develop a code for to print even numbers from 1 to 10 using for loop.

```
#include <stdio.h>
void  main()
{
   int i;
    for(i=1; i<=10; i++)
   {
          if(i%2 == 0)
             printf("%d\n", i);
        }
   }
```

1(E). Differentiate between break and exit statements.

break	**exit()**
break causes an immediate exit from the switch or loop (for, while or do).	exit() terminates program execution when it is called.
No header files needs to be included in order to use break statement in a C program.	stdlib.h needs to be included in order to use exit().
Example of break *// some code here before while loop* while(**true**) { ... if(condition) **break**; } *// some code here after while loop*	Example of exit() *// some code here before while loop* while(**true**) { ... if(condition) exit(-1); } *// some code here after while loop*

1(F). Develop a code for to copy one string to another string.

```
#include<stdio.h>
#include<string.h>
void main()
{
   char str1[20], str2[20];
   printf("Enter the string: ");
   gets(str1);
   strcpy(str2, str1);
   printf("\nString 2 = %s", str2);
    }
```

2.(A) Differentiate between increment and decrement operators.

S.No	Parameters	Increment Operators	Decrement Operators
1	Meaning and Definition	This type of operator adds value to a given operand.	This type of operator subtracts value from a given operand.
2	Prefix	During its pre-increment, we first increment the overall value of any variable. We then use it inside an expression.	In the case of a pre-decrement, we first decrement the overall value of a variable and then use it inside any given expression.
3	Postfix	In the case of post-increment, we first use the given value inside an expression and increment it later.	In the case of post-decrement, we first use the value inside an expression and decrement it later.
4	Uses	It helps in the process of looping and decision making	It also helps in the process of looping and making decisions.

2 (B). Demonstrate the input and output functions of C programming with example programs.

S.No	Input function	Output function
1	getchar(): to read a single character from keyboard. syntax: variable=getchar(); example: char ch; ch=getchar();	putchar(): to print a single character on the output screen. syntax: putchar(variable); example: char ch; ch=getchar(); putchar(ch);
2	gets() : to read a string from keyboard syntax: gets(string); example: char name[50]; gets(name);	puts() : to print a string on the output screen. syntax: puts(string); example: char name[50]; gets(name); puts(name);
3	scanf() : to read all types of values from keyboard. syntax: scanf("format characters",&variables); char ch; int a; float b; scanf("%c%d%f",&ch,&a,&b);	printf(): To print all types of variable values and statements on the output screen. Syntax: printf("("format characters",variables"); example: char ch; int a; float b; scanf("%c%d%f",&ch,&a,&b); printf(("%c%d%f",ch,a,b);

2(C). Develop a code for to print a transpose matrix of a given matrix.

```
#include <stdio.h>
void main() {
  int a[10][10], i,j,transpose[10][10], r, c;
  printf("Enter rows and columns: ");
  scanf("%d %d", &r, &c);
  printf("\nEnter matrix elements:\n");
  for (i = 0; i < r; ++i){
  for (j = 0; j < c; ++j) {
  scanf("%d", &a[i][j]);
  } }
for  (i = 0; i < r; ++i){
  for (j = 0; j < c; ++j) {
    transpose[j][i] = a[i][j];
  } }
  printf("\nTranspose of the matrix:\n");
  for (int i = 0; i < c; ++i){
  for (int j = 0; j < r; ++j) {
    printf("%d\t ", transpose[i][j]);
  }
 printf("\n");
} }
```

2(D). Develop a code for to print a reverse string of a given string.

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
int main()
{
   char str[100];
    clsrc();
   printf("enter a string: ");
   gets(str);
   strrev(str);
   printf("reversed string: %s", str);
   return 0;
}
```

3(A). Discuss about Data types in C programming.

There are various types of data types in C, which are:

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Range of Values of int Data Type in C

Data Type	Format Specifier	Range	Memory Size (in bytes)
Int	%d	-2,147,483,648 to 2,147,483,647	4
short int	%hd	-32,768 to 32,767	2
unsigned int	%u	0 to 4,294,967,295	4
unsigned short int	%hu	0 to 65,535	2
long int	%ld	-2,147,483,648 to 2,147,483,647	4
unsigned long int	%lu	0 to 4,294,967,295	4
long long int	%lld	-(2^63) to (2^63)-1	8
unsigned long long int	%llu	0 to 18,446,744,073,709,551,615	8

Basic Data Types in C

The fundamental data types in C, such as integers, float, characters, etc., are used to represent simple values and are known as the "basic data types."

1. int

The int data type is used to store integers (whole numbers) like -1, 0, 42, etc.

Syntax

```
int variable_name;
```

2. char

The char data type is used to store single characters like 'A', '1', or '$'. The character must be surrounded by single quotes and we must use the %c format specifier to print it.

Syntax

```
char variable_name;
```

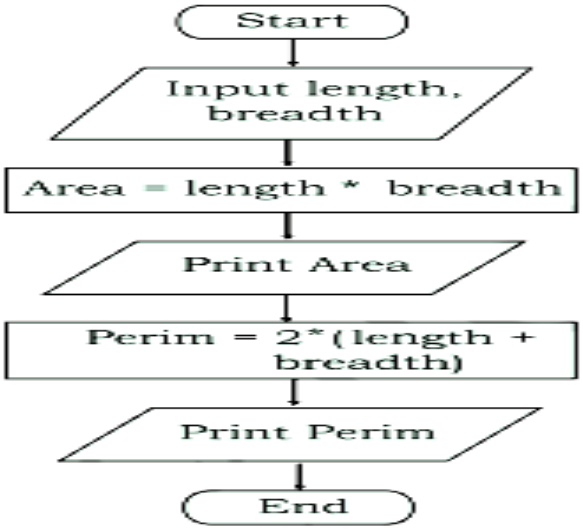
3. float

The float data type is used to store single-precision floating-point numbers, which can represent decimal values with limited precision. This data type size is 4 bytes, i.e., it occupies 4 bytes of memory. It ranges between +/- 3.4E-38F up until +/- 1 .7E+308F depending on the system's architecture.

Syntax

```
float variable_name;
```

3(B). Draw the flow chart for to find the area and perimeter of a rectangle.



4(A). Discuss about various type of operators.

An operator is a symbol used to perform arithmetic and logical operations in a program. That means an operator is a special symbol that tells the compiler to perform mathematical or logical operations. C programming language supports a rich set of operators that are classified as follows.

1. **Arithmetic Operators**
2. **Relational Operators**
3. **Logical Operators**
4. **Increment & Decrement Operators**
5. **Assignment Operators**
6. **Bitwise Operators**
7. **Conditional Operator**
8. **Special Operators**

Arithmetic Operators (+, -, *, /, %)

The arithmetic operators are the symbols that are used to perform basic mathematical operations like addition, subtraction, multiplication, division and percentage modulo

Relational Operators (<, >, <=, >=, ==, !=)

The relational operators are the symbols that are used to compare two values. That means the relational operators are used to check the relationship between two values. Every relational operator has two results TRUE or FALSE. In simple words, the relational operators are used to define conditions in a program.

Logical Operators (&&, ||, !)

The logical operators are the symbols that are used to combine multiple conditions into one condition.

Increment & Decrement Operators (++ & --)

The increment and decrement operators are called unary operators because both need only one operand. The increment operators adds one to the existing value of the operand and the decrement operator subtracts one from the existing value of the operand.

Pre-Increment or Pre-Decrement

In the case of pre-increment, the value of the variable is increased by one before the expression evaluation.

Post-Increment or Post-Decrement

In the case of post-increment, the value of the variable is increased by one after the expression evaluation.

Assignment Operators (=, +=, -=, *=, /=, %=)

The assignment operators are used to assign right-hand side value (Rvalue) to the left-hand side variable (Lvalue). The assignment operator is used in different variants along with arithmetic operators

Bitwise Operators (&, |, ^, ~, >>, <<)

The bitwise operators are used to perform bit-level operations in the c programming language. When we use the bitwise operators, the operations are performed based on the binary values.

Conditional Operator (?:)

The conditional operator is also called a **ternary operator** because it requires three operands. This operator is used for decision making. In this operator, first we verify a condition, then we perform one operation out of the two operations based on the condition result. If the condition is TRUE the first option is performed, if the condition is FALSE the second option is performed.

sizeof operator

This operator is used to find the size of the memory (in bytes) allocated for a variable

Comma operator (,)

This operator is used to separate variables while they are declaring, separate the expressions in function calls, etc.

4(B). Discuss about operator precedency and operator Associativity.

What is Operator Precedence?

Operator precedence is used to determine the order of operators evaluated in an expression

What is Operator Associativity?

Operator associativity is used to determine the order of operators with equal precedence evaluated in an expression.

In c programming language the operator precedence and associativity are as shown in the following table.

Operator	Description	Associativity
() [] . ->	Parentheses (grouping) Brackets (array subscript) Member selection via object name Member selection via pointer	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Unary preincrement/predecrement Unary plus/minus Unary logical negation/bitwise complement Unary cast (change *type*) Dereference Address Determine size in bytes	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right

&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
?:	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

In the above table, the operator precedence decreases from top to bottom and increases from bottom to top.

5(A). Develop a code for while loop and do while loop.

while loop:

```
#include<stdio.h>
void main()
{
int i;
i=1;
while(i<=10)
{
printf(“%d\n”,i);
i++;
}
}
```

do while loop:

```
#include<stdio.h>
void main()
{
int i;
i=1;
do{
printf(“%d\n”,i);
i++;
} while(i<=10);
}
```

5(B). Develop a code for find the even numbers sum and odd numbers sum of a given two-dimensional array.

```
#include <stdio.h>
int main() {
   int arr[rows][cols],rows, cols, i, j, evenSum = 0, oddSum = 0;
   printf("Enter number of rows and columns: ");
   scanf("%d%d", &rows,&cols);
   printf("Enter elements of the array:\n");
   for (i = 0; i < rows; i++) {
      for (j = 0; j < cols; j++) {
         printf("Element [%d][%d]: ", i, j);
         scanf("%d", &arr[i][j]);
          if (arr[i][j] % 2 == 0) {
            evenSum += arr[i][j];
         } else {
            oddSum += arr[i][j];
         }
      }
   }
   printf("Sum of even numbers: %d\n", evenSum);
   printf("Sum of odd numbers: %d\n", oddSum);
   return 0;
}
```

6(A). Demonstrate any five string handling functions.

In C, string handling functions are provided by the standard library <string.h> to manipulate strings efficiently. Here are five commonly used string handling functions along with a C program to demonstrate their usage:

1. strlen() - String Length

This function returns the length of the string (excluding the null character \0).

2. strcpy() - String Copy

This function copies one string into another.

3. strcat() - String Concatenation

This function concatenates (appends) one string to the end of another.

4. strcmp() - String Comparison

This function compares two strings lexicographically. It returns:

- 0 if the strings are equal,
- A negative value if the first string is lexicographically smaller,
- A positive value if the first string is lexicographically larger.

5. strrev() - String Reverse

This function reverses the given string. Note that strrev() is not part of the C standard library, but it's available in some compilers. If strrev() is not available in your compiler, you can manually write a function to reverse the string.

6(B). **Develop a code for nested if statement and switch statement.**

C Program with Nested if:

```
#include <stdio.h>
void main() {
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);
   if (num > 0) {
      printf("The number is positive.\n");
      if (num % 2 == 0) {
         printf("The number is even.\n");
      } else {
         printf("The number is odd.\n");
      }
   } else if (num < 0) {
      printf("The number is negative.\n");
      if (num % 2 == 0) {
         printf("The number is even.\n");
      } else {
         printf("The number is odd.\n");
      }
   } else {
      printf("The number is zero.\n");
       }
```

C Program Using Switch Statement:

```
#include <stdio.h>
void main() {
   int choice;
   printf("Enter a number (1 to 3): ");
   scanf("%d", &choice);
   switch (choice) {
      case 1:
         printf("You chose option 1.\n");
         break;
      case 2:
         printf("You chose option 2.\n");
         break;
      case 3:
         printf("You chose option 3.\n");
         break;
      default:
         printf("Invalid choice! Please enter a number between 1 and 3.\n");
         break;
   } }
```