1) Explain the different types of C constants and the rules for constructing integer, real and character constants with examples.

⇒ C constants are fixed values that do not change during the execution of a program. Here are the main types of C constants:

i) integer constants
- Represent whole numbers without a fractional part.
- Examples : 10, -5, 0
- Types : Decimal, octal, Hexadecimal

ii) Real (floating-point) constants
- Represent numbers with a fractional part or exponential form.
- Examples : 3.14, -0.001, 6.022e23

iii) Character constants
- Represent a single character enclosed in single quotes.
- Examples : 'A', 'g', '#'

iv) String constants
- Represent sequences of characters enclosed within double quotes.
- Example : "Hello", "C programming"

Rules for constructing Integer constants

1) Must contain Digits only.
- Integer constants consist of digits from 0 to 9.

2) No Decimal point:
- Integer constants should not have a decimal point.

3) Positive or Negative:
- They can be preceded by a positive (+) or negative (-) sign, or no sign at all.

4) Three types of Integer constants:
- Decimal (Base 10): Composed of digits from 0 to 9.
  Examples: 123, -456
- Octal (Base 8): Begins with 0 and use digits from 0 to 7
  Examples: 012 (equivalent to decimal 10)
- Hexadecimal (Base 16) : Begins with 0x or 0X and uses digits 0-9 and letters A-F.
  Examples: 0x1A (equivalent to decimal 26)

5) No Spaces or special characters:
- Integer constants cannot contain commas, spaces, or special symbol like $.

Example:

```
#include<stdio.h>
int main(){
  int a = 123;    // Decimal constant
  int b = -456;   // Negative decimal constant
  int c = 051;    // octal constant
  int d = 0X1A;   // Hexadecimal constant

printf("integer constants : %d %d %d %d \n", a, b, c, d);
  return 0;
  }
```

Rules for constructing Real (Floating - point) constants.

1) Must contain a Decimal point;

- Real constants must have a fractional part, i.e a decimal point.

Example : 3.14

2) can B in exponential form;

- Real constants can also be written in exponential notation.

Example: 1.23e4 (equivalent to 1.23 x 10^4)

3) can be positive or negative.

- Real constants can be preceded by + or -.

examples :-0.001 , +23.45

4) must have at least one digit.
- At least one digit must be present before and after the decimal point in a real constant.

Example : 0.5

5) No Spaces or special characters:
- Real constants cannot contain spaces, commas, or symbols like #.

Example:

```
#include <stdio.h>
int main ()
{
  float a = 3.14;    // Simple real constant
  float b = -0.001;  // Negative real constant
  float c = 2.5e3;   // Exponential form

printf("constants : %f %f %f \n", a, b, c);
return 0;
}
```

Rules for constructing character constants

1) Enclosed in Single Quotes:
- Character constants must be enclosed within single quotes

Example : 'A'

2) Single character only:
- only one character, digit or special symbol is allowed within the quotes
Example : '7', '#'

3) Escape sequences:
- Special characters can be represented using escape sequences starting with a backslash (\).
Examples : '\n' (newline), '\t' (tab), '\0' (null character)

4) Case sensitivity:
- Character constants are case-sensitive meaning 'A' and 'a' are different.

Program:

```
#include <stdio.h>
int main()
{
  Char  Ch1  = 'A';
  Char  ch2  = 'g';
  Char  ch3  = '#';

printf("Character Constant = %c %c %c \n", Ch1 ch2, Ch3);
return 0;
}
```

2) What are variables in C? Discuss the types of variables and the rules of constructing valid variable names in C.

→ Variables are the containers that is used to ~~share~~ store values. It's storage place where same form of data is stored. Different variables required different amount of memory to store the data.

Types of variables in C

I) local variables

- Definition: variables declared inside a function or block. They are only accessible within that function or block.
- Scope: Limited to the function or block where they are declared
- Lifetime: Exists only during the execution of the function or block.

Program:

```
#include<stdio.h>
int main(){
  int localvar=10;
printf("Local variable =%d", localvar);
return 0; }
```

Explanation: local var is a local variable that exists only inside the main function.

II) Global variables

- Definition : variables declared outside of all functions. They can be accessed by any function in the program.
- Scope : Throughout the entire program.
- Lifetime : Exists until the program ends.

Program:

```
#include<stdio.h>
int main() {
int globalvar = 20;
printf("Global variable = %d", globalvar);
return 0;
}
```

Explanation: globalvar is accessible inside main because it is declared outside any function.

III) Static variables

- Definition : variables declared with the static keyword they retain their values between function calls.
- Scope : local to the function or block but retain their value across multiple calls.
- Lifetime : Exists for the entire duration of the program.

Program :

```
#include<stdio.h>
int main() {
static int staticvar = 30;
```

```
    printf("Static variable : %d", staticVar);
    staticVar++;  // modifies staticVar for the next call (if any)
    return 0;
}
```

Explanation: staticVar retains its value even after the function ends, which is seen if called multiple times.

Rules for constructing variable Names

A variable can have a short name like x and y or a more descriptive name like age, price etc.

Variable naming rules:

- A variable name must start with a letter or an underscore character (_)
- A variable name cannot start with a digit.
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9 and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- There is no limit on the length of the variable name.
- A variable name cannot contain spaces
- The variable name cannot be any keywords.

Multi-word variable Names

- Variable names with more than one word can be difficult to read.
- There are several techniques you can use to make them more readable:

Camel case
- Each word, except the first, starts with a capital letter:
- myVariableName = "John"

Pascal case
- Each word starts with a capital letter eg: MyVariableName = "John"

Snake case
- Each word is separated by underscore. eg: MY_name = "John"

3) Describe the process of type conversion in assignments in C. Provide examples to demonstrate both implicit and explicit type conversion.

⇒ Type conversion is the process of converting a variable from one data type to another. This process, also known as type casting, can happen either implicitly or explicitly.

i) Implicit conversion (Automatic type conversion)

- The compiler automatically converts one data type to another without any explicit instruction from the programmer.
- occurs when a smaller data type is assigned to a larger data type

Example:

float

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

   b=a;
printf ("value of a (int) : %d\n", a);
printf ("value of b (float): %f\n", b);
return 0;
}
```

ii) Explicit conversion (Type casting):

- The programmer explicitly defines the conversion using casting.
- This is used when the conversion could result in data loss or when converting between incompatible types.

example:

```
#include<stdio.h>
int main()
{
float a=5.75;
int b;
b= (int)a;

printf("value of a (float): %f\n", a);
printf("value of b (int) after explicit conversion : %d\n", b);
return 0;
}
```

4) Compare the different data types available in C, including integers (long & short), chars (signed & unsigned), and floating-point types (float & double).

→ Data types are fundamental aspects of programming that define the type of data a variable can hold. Some basic data types in C.

i) Integers

Integers are whole numbers without any fractional or decimal part. They are used to store numerical values such as counts, indexes, or any whole-number calculations.

Types:

- int: Standard integer type.
- Size: Typically 2 or 4 bytes depending on the system, capable of holding values from -32,768 to 32,767 (for 2 bytes) or -2,147,648 to 2,147,483,647 (for 4 bytes)
- int age = 25; // Declares an integer variable 'age' with a value of 25.

▲ Long integer

- Definition: A longer version of the int type that can store large numbers
- Size: Typically 4 or 8 bytes.
- Range: Larger ranges than int: 2,147,483,648 to 2,147,483,647 or beyond 8 bytes

Example: longpopulation = 100000000;

▲ Short Integer

Definition : A shorter version of the int type that uses less memory.

- Size : usually 2 bytes.
- Range: -32, 768 to 32, 767.

Example

```
Short Score = 3000;
```

2) Chars (Characters)

used to store single characters or small integers. Each character is stored as an integer according to its ASCII value.

- Size : 1 byte
- Range : Typically -128

▲ Signed char : can store negative and positive character values

▲ unsigned char : Stores only positive values

```
Char letter = 'A';  // Stores character 'A'

Signed char SingleLetter = -65; // Stores negative ASCII values

Unsigned char age = 255; // Stores maximum unsigned value
```

iii) Floating - point type

- Are used to represent real numbers that have fractional parts.

▲ Float:

- Definition: used for storing single - precision floating - point numbers (decimals)
- Size = 4 bytes
- Range : Approximately 3.4E-38 to 3.4E +38.

example

float price = 19.99

▲ Double

- Definition: used for storing double - precision floating - point numbers ; which allows more precision and a large range.
- Size : 8 bytes.
- Rang : Approximately 1.7E-308 to 1.7E+308.

example

double preciseValue = 12345.6789

5) What is the purpose of comments in a C program? Explain the different types of comments and how they improve the code readability with examples.

-> Comments can be used to explain the code and to make it more readable. compiler ignores the comments during compilation. Comments are useful to make the code more understandable for other as they explain what certain parts of the code are meant to do.

There are two types of comments:

i) Single line comments

- Single line comments starts with two forward slashes (//).
- Any text between // and the end of the line is ignored by the compiler (will not executed)

Example:

```
#include<stdio.h>
int main(){
  int a;        // This comment explain i.e a variable
printf("Enter a number:"); // Display message
scanf("%d", &a); // storing the input of user
printf("%d", a); // printing the same value

}
```

ii) Multi-line comment

- Multi-line comments starts with /* and ends with */.
- Any text between /* and */ will be ignored by the compiler

Example:

```
#include<stdio.h>
int main() {
/*  int a, b, c;
    printf("Enter 1st number");
    scanf("%d", &a);
    Printf("\n Enter 2nd number");
    scanf("%d", &b);
     c= a+b;
    printf("The sum = %d", &c);      */

    Printf("Hello world");
```

In the above program only Hello world is printed because other codes are commented out. This can be done to close some operations in the program.