C Programming QB Module 1 Part A Solutions

@Ujjwal Acharya

1. **List the major components of computer**

> The major components of a computer include:
> 1. Memory
> 2. Processor
> 3. I/O Devices
> 4. Storage and
> 5. Operating system

2. **Define the term operating system.**

> The operating system provides services such as a user interface, file and database access, and interfaces to communication systems such as Internet protocols. The primary purpose of this software is to keep the system operating in an efficient manner while allowing the users access to the system.

3. **Define the term algorithm.**

> The Algorithm is a finite sequence of instructions, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time. No matter what the input values may be, an algorithm terminates after executing a finite number of instructions. We represent an algorithm using a pseudo-language that is a combination of the constructs of a programming language together with informal English statements. The ordered set of instructions required to solve a problem is known as an algorithm.

4. **Define the term flowchart.**

> Flowchart is a diagrammatic representation of an algorithm. Flowchart is very helpful in writing program and explaining program to others. Different symbols are used for different states in flowchart. For example: Input/Output and
> decision making has different symbols.

5. **Write the properties of an algorithm.**

> The properties of a good algorithm are:
> 1. Precision – the steps are precisely stated (defined).
> 2. Uniqueness – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
> 3. Finiteness – the algorithm stops after a finite number of instructions are executed.

> 4. Input – the algorithm receives input.
> 5. Output – the algorithm produces output.
> 6. Generality – the algorithm applies to a set of inputs.

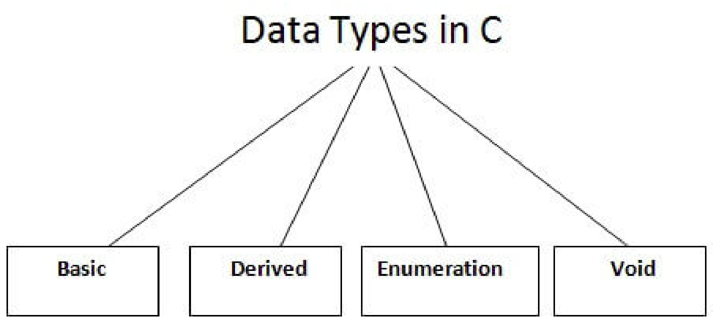
6. Write how a compiler works.

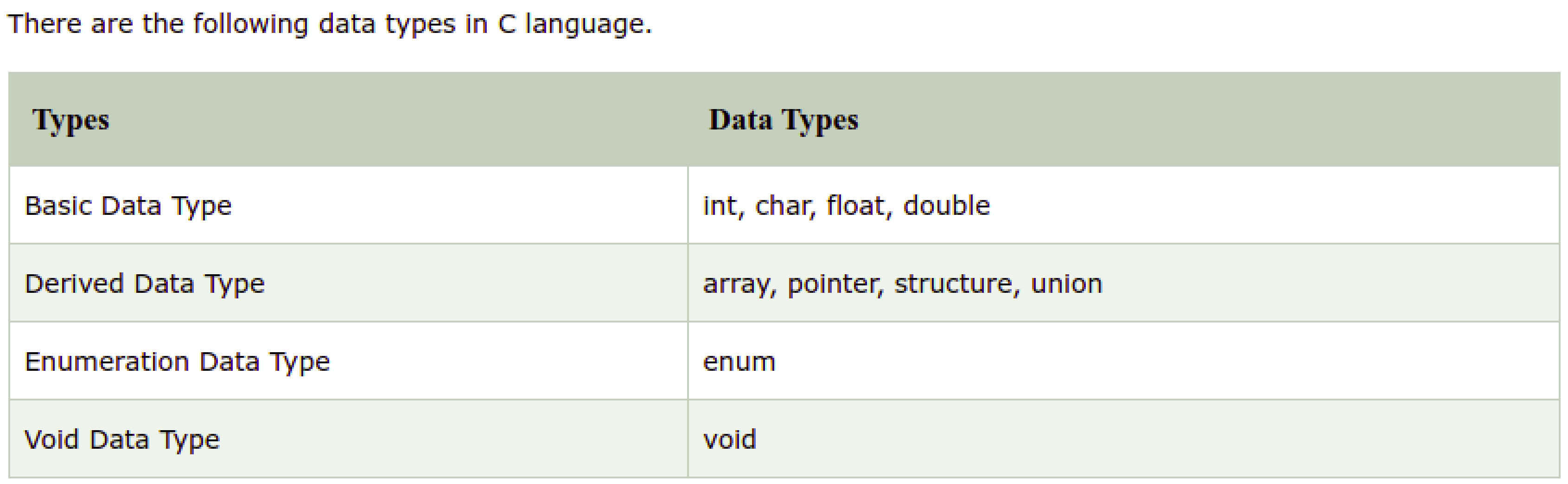
> A Compiler is used to compile an entire program and an executable program is generated through the object program. It translates the source code written in a high-level language into the corresponding object code of the low-level language, therefore making it accessible for the computer to process the information. This translation process is called compilation and the entire high level program is converted into the executable machine code file.

7. Compare the differences between compiler and an interpreter.

A COMPILER	B Interpreter
1. A Compiler is used to compile an entire program and an executable program is generated through the object program	1. An interpreter is used to translate each line of the program code immediately as it is entered
2. The executable program is stored in a disk for future use or to run it in another computer	2. The executable program is generated in RAM and the interpreter is required for each run of the program
3. The compiled programs run faster	3. The Interpreted programs run slower
4. Most of the Languages use compiler	4. A very few languages use interpreters

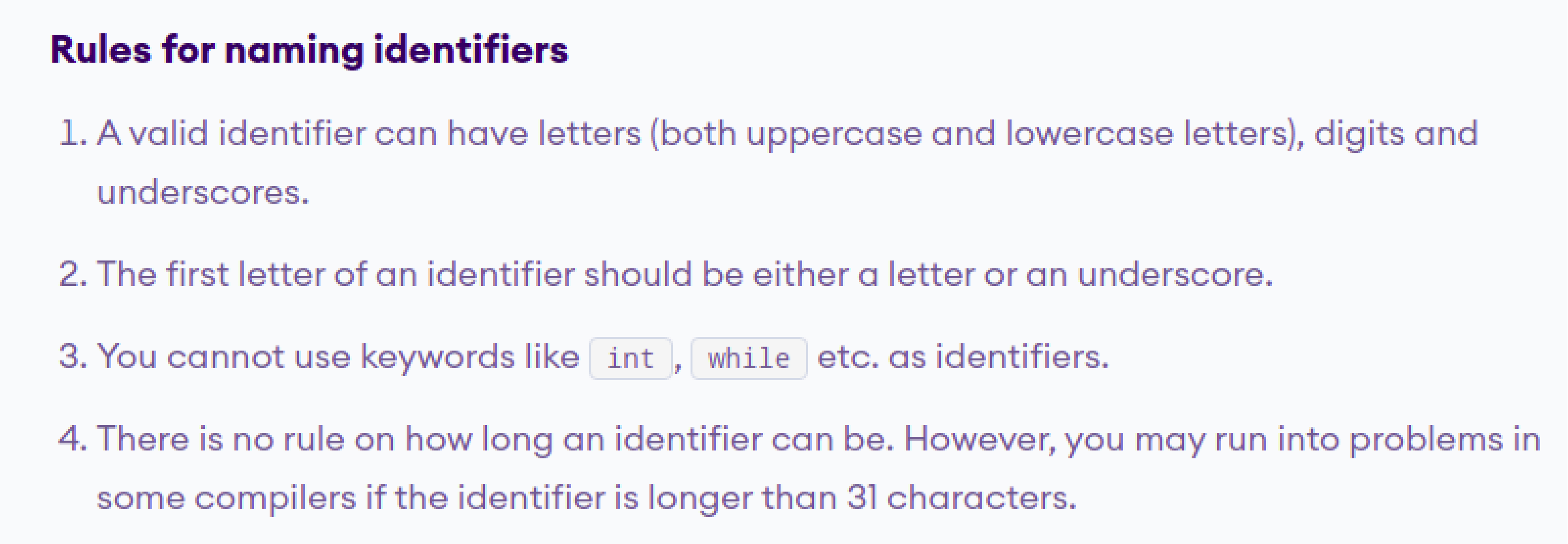
8. Write about datatypes in C.




There are the following data types in C language.

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

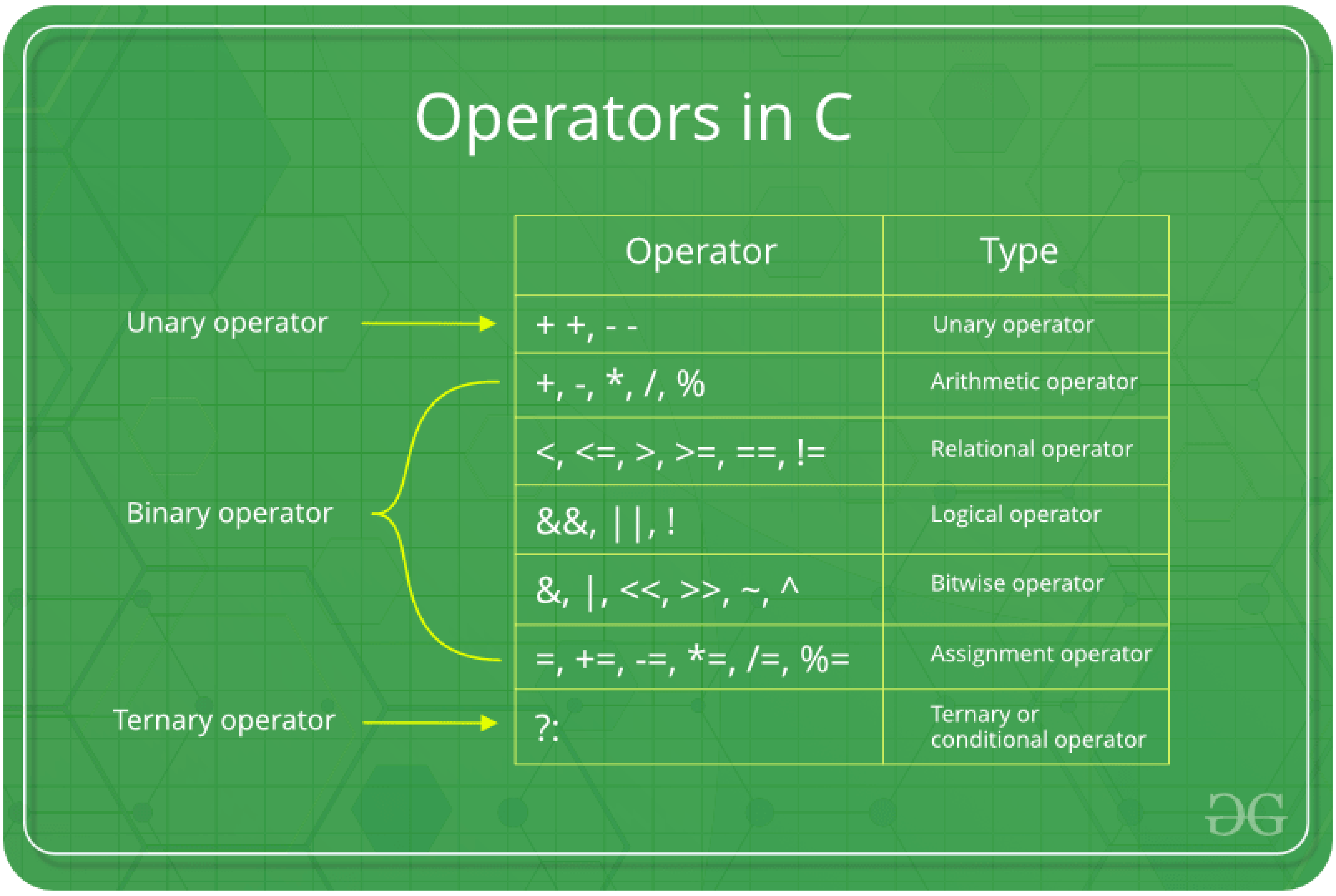
9. **List the rules for naming identifiers in C.**



Rules for naming identifiers

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
2. The first letter of an identifier should be either a letter or an underscore.
3. You cannot use keywords like `int`, `while` etc. as identifiers.
4. There is no rule on how long an identifier can be. However, you may run into problems in some compilers if the identifier is longer than 31 characters.

10. **List the types of operators in C.**



Operators in C

Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, | |, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

Unary operator
Binary operator
Ternary operator

11. **Explain operator precedence in C**

> Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others. For example, the multiplication operator has a higher precedence than the addition operator.
> For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

12. **Compare & and * operators in C.**

> "*" Operator is used as pointer to a variable. Example: * a where * is pointer to the variable a.

> "&" Operator is used to get the address of the variable. Example: &a will give address of a.

13. Find the output of the following program.

```
#include <stdio.h>

void main()
{
 1 < 2 ? return 1 : return 2;
 }
```

```
#include <stdio.h>
void main()
{
 (1 < 2)? return 1 : return 2;
 }
//The Ternary operator accepts conditions and not statements. Since the return is a statement and not a condition, the above program will return an error
```

14. Find the output of the following program.

```
#include <stdio.h>

void main()
{
 printf("value is = %d",(10++));
}
```

```
#include <stdio.h>
void main()
{
 printf("value is = %d",(10++));
}
//This code is returning Compilation Error
```

15. Find the output of the following program.

```
#include <stdio.h>

void main()
{
const char var='A';
++var;
printf("%c",var);
}
```

```
#include <stdio.h>
void main()
{
const char var='A';
++var;
printf("%c",var);
}
//var here is a constant so the compiler will not iterate the characters
```

16. Find the output of the following program.

```
#include <stdio.h>

void main()
{
```

```
int x=(20 || 40 ) && (10);
printf("x= %d",x);
}
```

```
#include <stdio.h>
void main()
{
int x=(20 || 40 ) && (10);
printf("x= %d",x);
}
//Output: x= 1
```

17. Find the output of the following program.

```
#include <stdio.h>

int main()
{
 int i;
 i = 1, 2, 3;
 printf("%d", i);
 return 0;
}
```

```
#include <stdio.h>
int main()
{
 int i;
 i = 1, 2, 3;
 printf("%d", i);
 return 0;
}
//Output: 1
```

18. Find the Output of thee following program.

```
#include <stdio.h>

void main()
{
 int a=3,b=2;
 a=a==b==0;
 printf("%d,%d",a,b);
}
```

```
#include <stdio.h>
void main()
{
 int a=3,b=2;
 a=a==b==0;
 printf("%d,%d",a,b);
}
//Output: 1,2
```

19. Find the Output of thee following program.

```
#include <stdio.h>

int main()
{
 float a;
 (int)a= 10;
 printf("value of a=%d",a);
```

return 0;
}

```
#include <stdio.h>
int main()
{
 float a;
 (int)a= 10;
 printf("value of a=%d",a);
 return 0;
}
//Error as Line 5 is not valid
```

20. Find the Output of thee following program.

#include <stdio.h>

int main()
{
int x = 2;
(x & 1)? printf("true") : printf("false");
return 0;
}

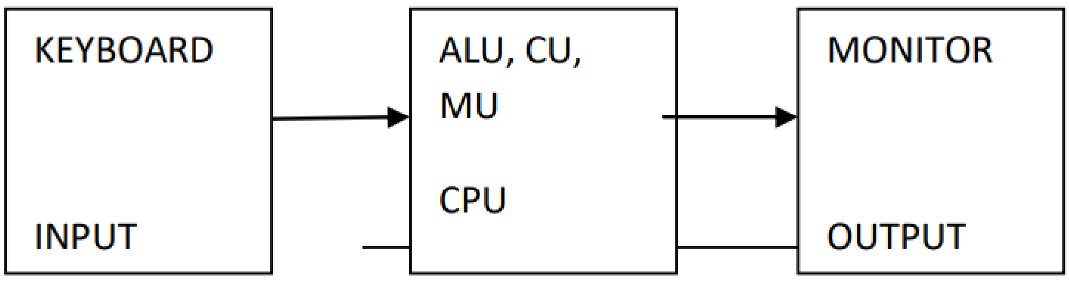
```
#include <stdio.h>
int main()
{
int x = 2;
(x & 1)? printf("true") : printf("false");
return 0;
}
//Output: false
```

C Programming QB Module 1 Part B Solutions

@Ujjwal Acharya

1. Explain in detail about computer hardware and software.

> **Computer Hardware:** The hardware of a computer system can be referred to as anything which we can touch and feel. Example: Keyboard and Mouse. The hardware of a computer system can be classified as - Input Devices, Processing Devices, and Output Devices.



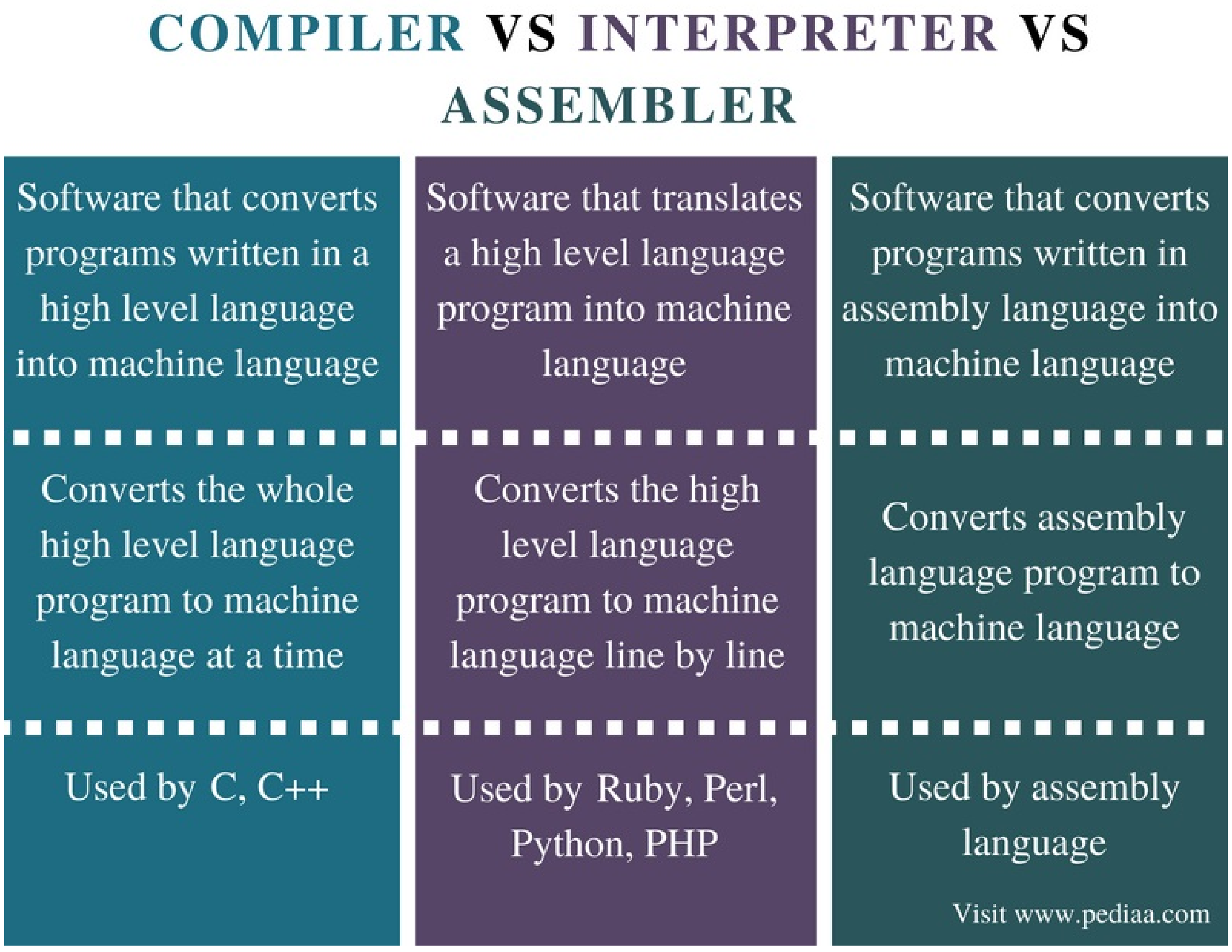
> **Computer Software:** The software of a computer system can be referred to as anything which we can feel and see. Example: Windows, icons
> Computer software is divided into two broad categories: system software and application software.
> System software manages the computer resources. It provides the interface between the hardware and the users. Application software, on the other hand, is directly responsible for helping users solve their problems.
> System software consists of programs that manage the hardware resources of a computer and perform required information processing tasks. These programs are divided into three classes: the operating system, system support, and system development.

2. Differentiate among high level, low level and middle level language.

> **High-Level Language:**
> → They are easy to learn and programs may be written in these languages with much less effort.
> → However, the computer cannot understand them and they need to be translated into machine language with the help of other programs known as Compilers or Translators. Like C++, Java, Python, etc.
> **Low-Level Language:**
> → The language whose design is governed by the circuitry and the structure of the machine is known as the Low Level / Machine language.
> → This language is difficult to learn and use. It is specific to a given computer and is different for different computers i.e. these languages are machine-dependent.
> → These languages have been designed to give better machine efficiency, i.e. faster program execution. Such languages are also known as Low-Level Languages.

> **Middle-Level Language:**
> → It bridges the gap between machine-understandable machine-level language and more conventional high-level language. This programming helps in writing system programming as well as application programming. Because C language has both the feature of high level and low-level language so it is often called middle level language.

3. Differentiate among compiler, assembler and interpreter.



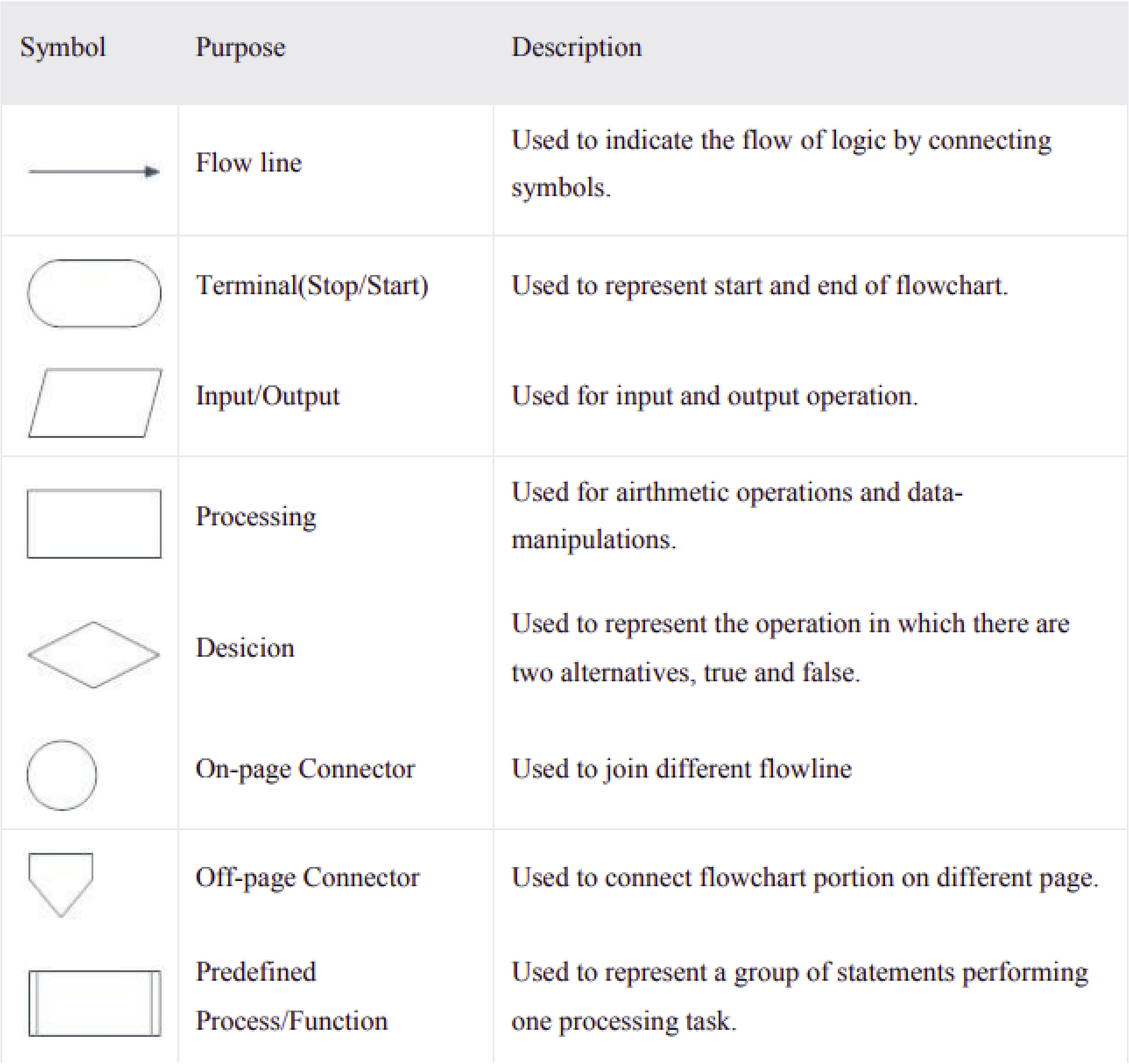
COMPILER VS INTERPRETER VS ASSEMBLER

Software that converts programs written in a high level language into machine language	Software that translates a high level language program into machine language	Software that converts programs written in assembly language into machine language
Converts the whole high level language program to machine language at a time	Converts the high level language program to machine language line by line	Converts assembly language program to machine language
Used by C, C++	Used by Ruby, Perl, Python, PHP	Used by assembly language

Visit www.pediaa.com

4. Define flowchart and explain different symbols used for constructing flowchart.

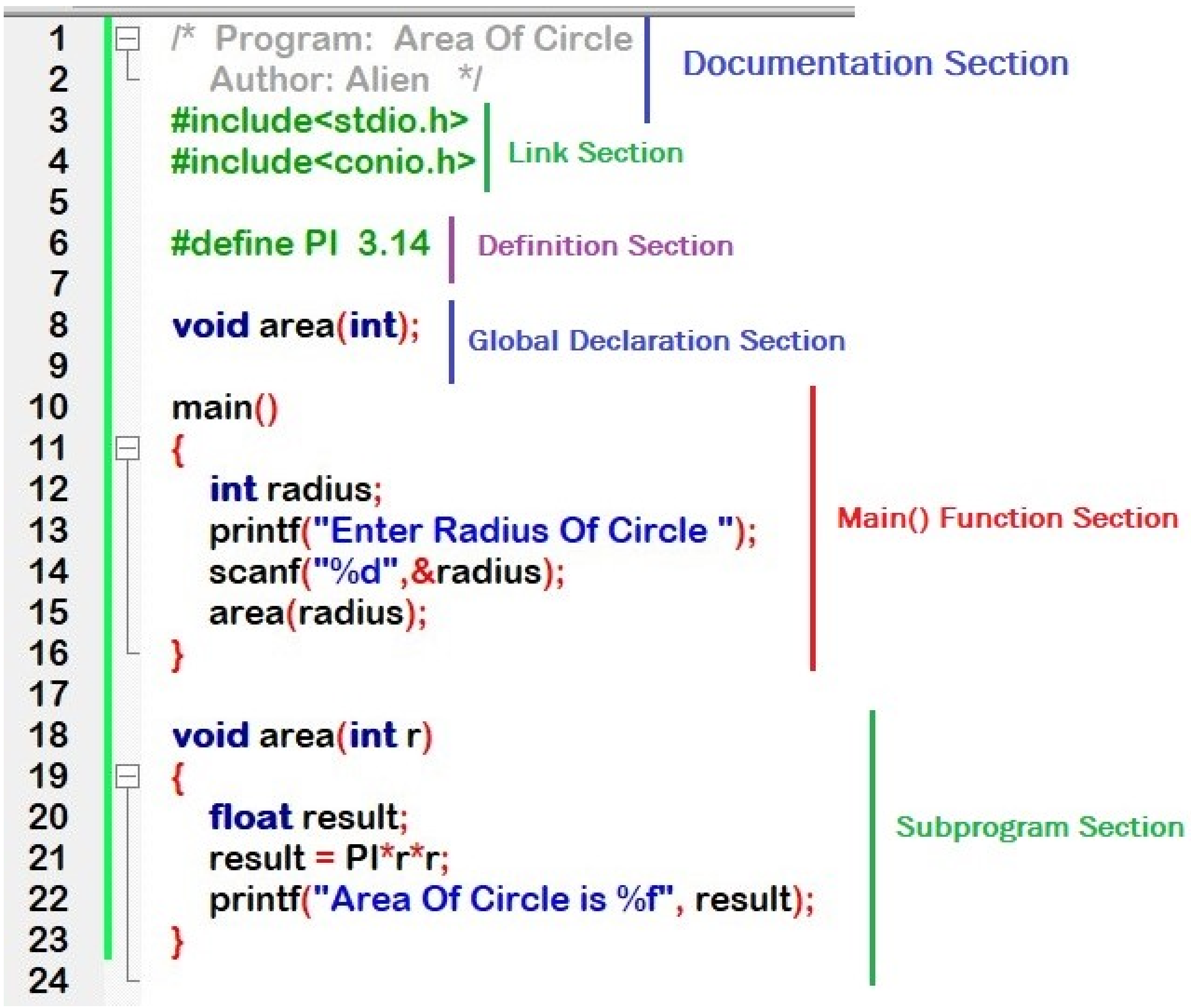
> A f**lowchart** is a diagrammatic representation of an algorithm. A flowchart is very helpful in writing programs and explaining programs to others.
> Different symbols are used for different states in a flowchart, For example, Input/Output and
> decision making has different symbols. The table below describes all the symbols that are used in making a flowchart.



Symbol	Purpose	Description
→	Flow line	Used to indicate the flow of logic by connecting symbols.
⬭	Terminal(Stop/Start)	Used to represent start and end of flowchart.
▱	Input/Output	Used for input and output operation.
▭	Processing	Used for airthmetic operations and data-manipulations.
◇	Desicion	Used to represent the operation in which there are two alternatives, true and false.
○	On-page Connector	Used to join different flowline
⬠	Off-page Connector	Used to connect flowchart portion on different page.
▭	Predefined Process/Function	Used to represent a group of statements performing one processing task.

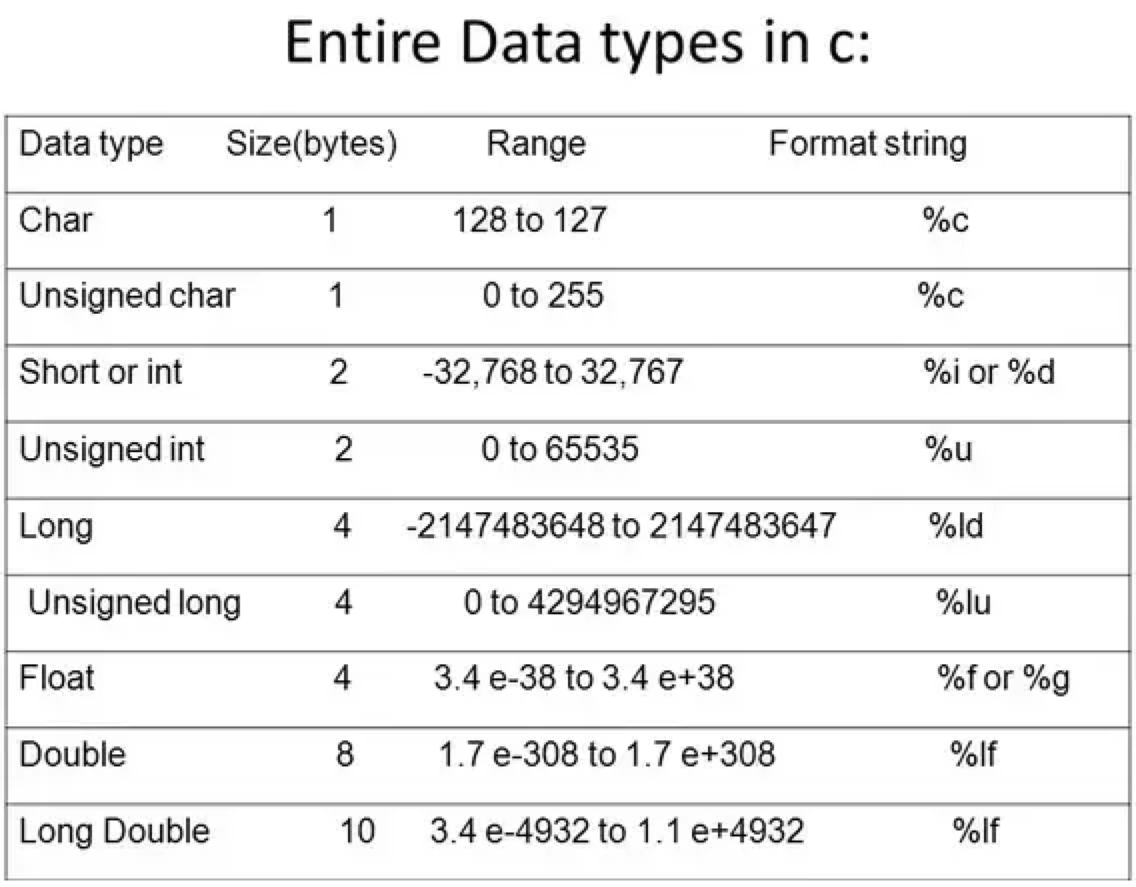
5. **Explain structure of a C program with example.**

> **Documentation section:** The documentation section consists of a set of comment lines giving the name of the program, the author, and other details, which the programmer would like to use it later.
> **1. Link section:** The link section provides instructions to the compiler to link functions from the system library such as using the #include directive.
> **2. Definition section:** The definition section defines all symbolic constants such as using the #define directive.
> **3. Global declaration section:** There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.
> **4. main () function section:** Every C program must have one main function section. This section contains two parts; the declaration part and the executable part
> **4.1. Declaration part:** The declaration part declares all the variables used in the executable part.
> **4.2. Executable part:** There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. All statements in the declaration and executable part end with a semicolon.
> **5. Subprogram section:** If the program is a multi-function program then the subprogram section contains all the user-defined functions that are called in the main () function.
> **6. User-defined functions** are generally placed immediately after the main () function, although they may appear in any order.



```
/* Program:  Area Of Circle
   Author: Alien   */
#include<stdio.h>
#include<conio.h>

#define PI  3.14

void area(int);

main()
{
   int radius;
   printf("Enter Radius Of Circle ");
   scanf("%d",&radius);
   area(radius);
}

void area(int r)
{
   float result;
   result = PI*r*r;
   printf("Area Of Circle is %f", result);
}
```

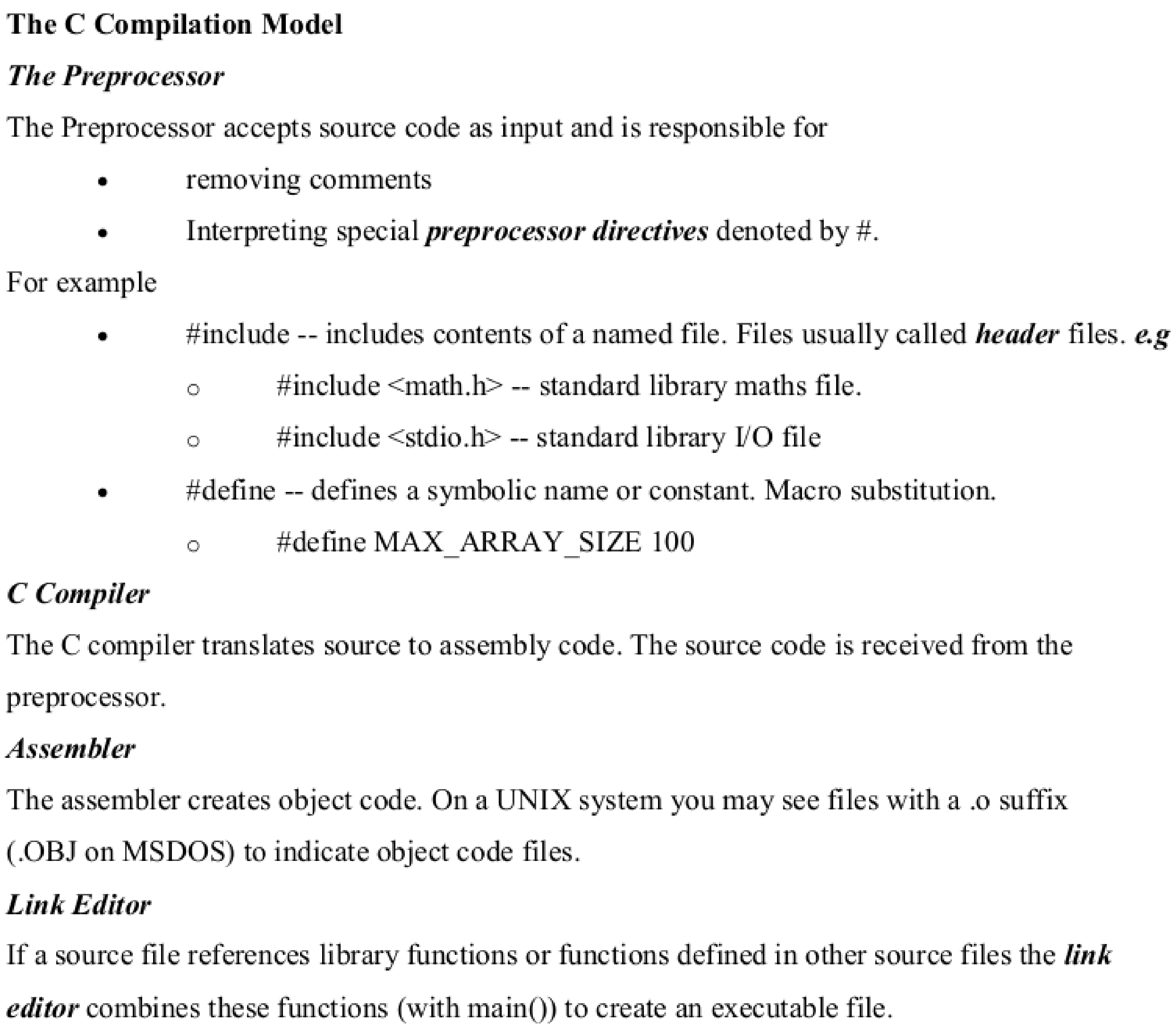
6. Explain all the data types with their ranges, examples.



Entire Data types in c:

Data type	Size(bytes)	Range	Format string
Char	1	128 to 127	%c
Unsigned char	1	0 to 255	%c
Short or int	2	-32,768 to 32,767	%i or %d
Unsigned int	2	0 to 65535	%u
Long	4	-2147483648 to 2147483647	%ld
Unsigned long	4	0 to 4294967295	%lu
Float	4	3.4 e-38 to 3.4 e+38	%f or %g
Double	8	1.7 e-308 to 1.7 e+308	%lf
Long Double	10	3.4 e-4932 to 1.1 e+4932	%lf

7. Explain Process of compiling and running a C program.



The C Compilation Model

The Preprocessor

The Preprocessor accepts source code as input and is responsible for

- removing comments
- Interpreting special ***preprocessor directives*** denoted by #.

For example

- #include -- includes contents of a named file. Files usually called ***header*** files. ***e.g***
  - #include <math.h> -- standard library maths file.
  - #include <stdio.h> -- standard library I/O file
- #define -- defines a symbolic name or constant. Macro substitution.
  - #define MAX_ARRAY_SIZE 100

C Compiler

The C compiler translates source to assembly code. The source code is received from the preprocessor.

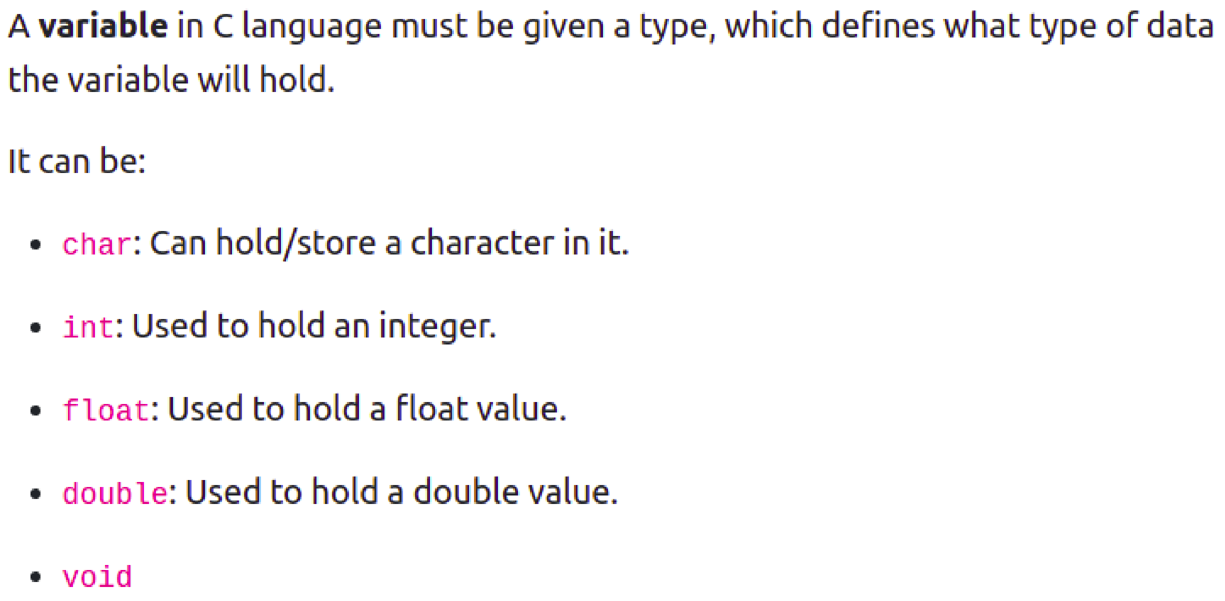
Assembler

The assembler creates object code. On a UNIX system you may see files with a .o suffix (.OBJ on MSDOS) to indicate object code files.

Link Editor

If a source file references library functions or functions defined in other source files the ***link editor*** combines these functions (with main()) to create an executable file.

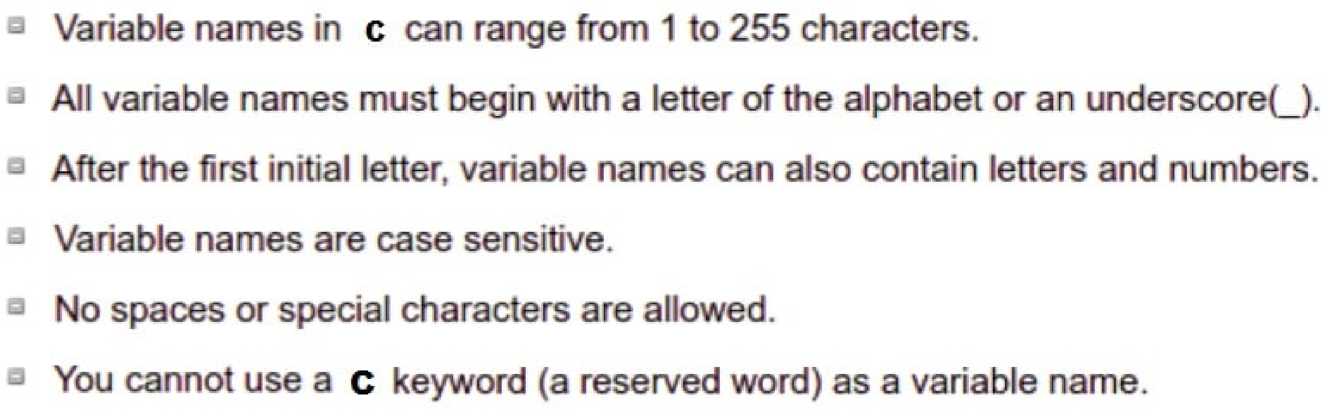
8. What is variable? Give the rules for variable declaration.



A **variable** in C language must be given a type, which defines what type of data the variable will hold.

It can be:

- `char`: Can hold/store a character in it.
- `int`: Used to hold an integer.
- `float`: Used to hold a float value.
- `double`: Used to hold a double value.
- `void`



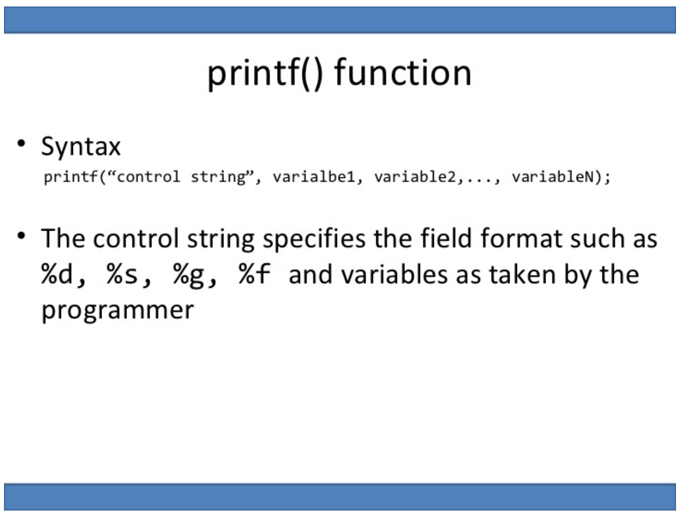
- Variable names in **C** can range from 1 to 255 characters.
- All variable names must begin with a letter of the alphabet or an underscore(_).
- After the first initial letter, variable names can also contain letters and numbers.
- Variable names are case sensitive.
- No spaces or special characters are allowed.
- You cannot use a **C** keyword (a reserved word) as a variable name.

Variable Declaration in C

A variable declaration provides assurance to the compiler that there exists a variable with the given type and name so that the compiler can proceed for further compilation without requiring the complete detail about the variable. A variable definition has its meaning at the time of compilation only, the compiler needs actual variable definition at the time of linking the program.

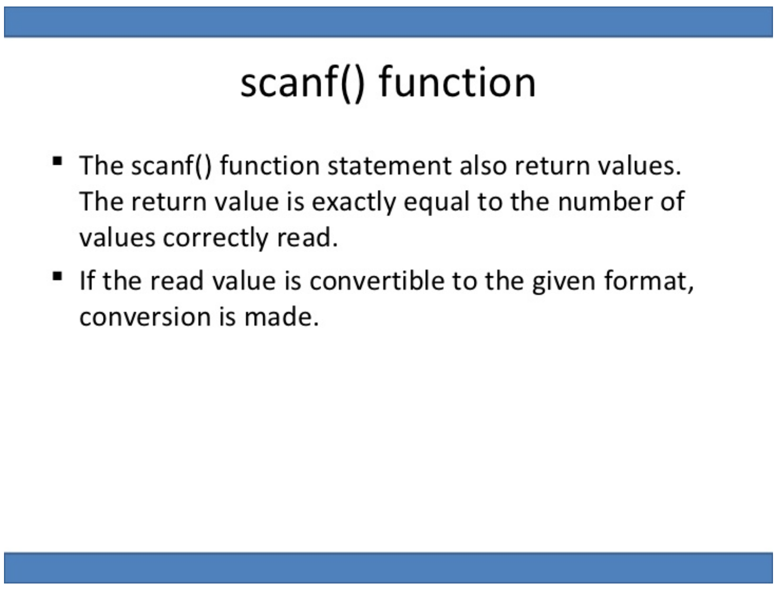
A variable declaration is useful when you are using multiple files and you define your variable in one of the files which will be available at the time of linking of the program. You will use the keyword **extern** to declare a variable at any place. Though you can declare a variable multiple times in your C program, it can be defined only once in a file, a function, or a block of code.

9. Explain syntax with examples of printf() and scanf() statements.



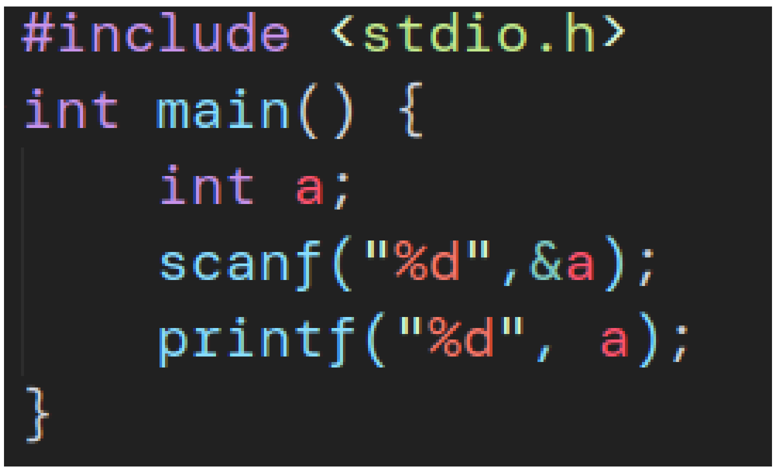
printf() function

- Syntax
  `printf(“control string”, varialbe1, variable2,..., variableN);`
- The control string specifies the field format such as `%d, %s, %g, %f` and variables as taken by the programmer



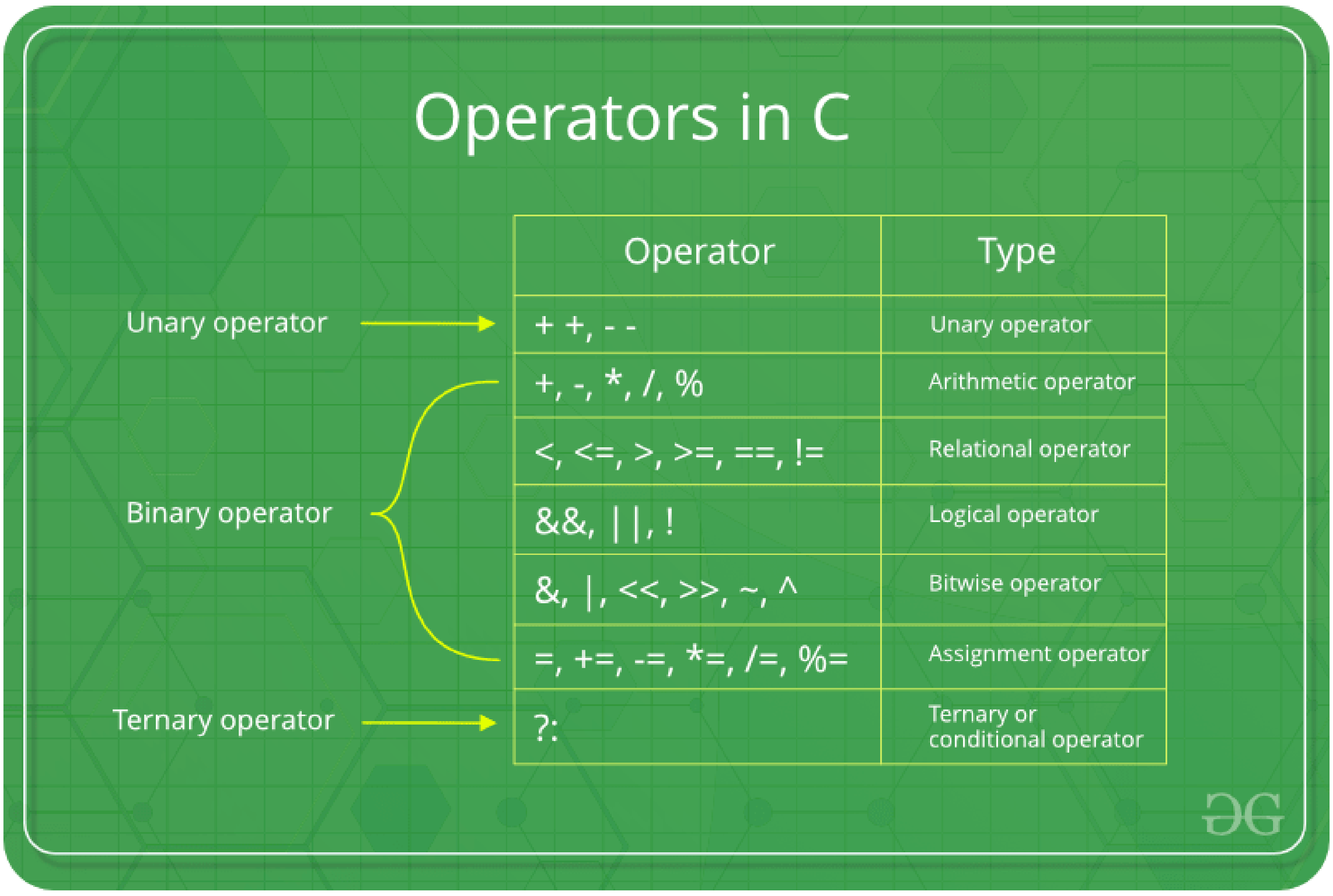
scanf() function

- The scanf() function statement also return values. The return value is exactly equal to the number of values correctly read.
- If the read value is convertible to the given format, conversion is made.



```
#include <stdio.h>
int main() {
    int a;
    scanf("%d",&a);
    printf("%d", a);
}
```

10. Explain in detail about the types of operators in C.



Operators in C

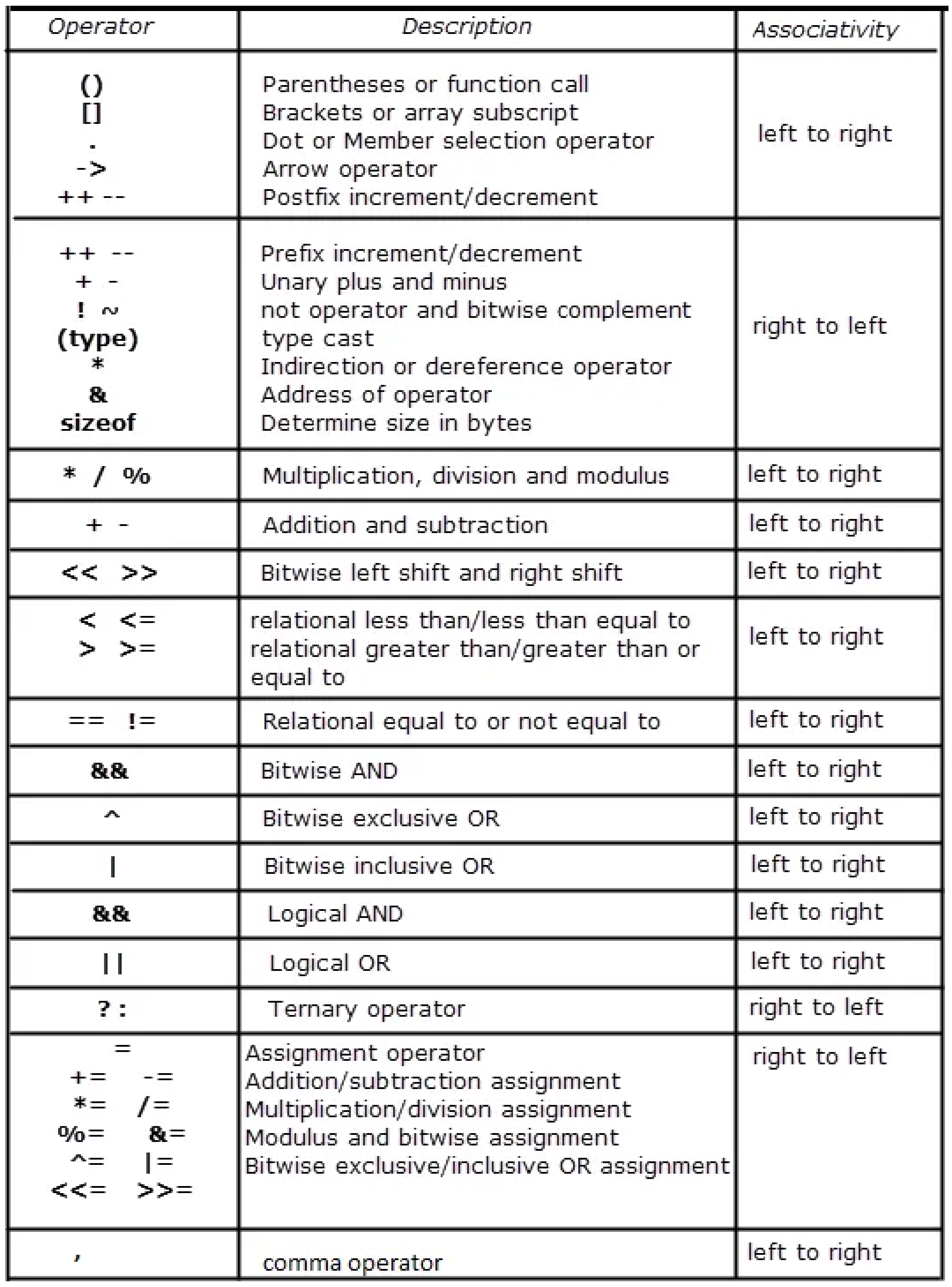
Unary operator

Binary operator

Ternary operator

Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, | |, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

11. Explain in detail about operator precedence and associativity in C.



Operator	Description	Associativity
() [] . -> ++ --	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left
* / %	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <= > >=	relational less than/less than equal to relational greater than/greater than or equal to	left to right
== !=	Relational equal to or 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 operator	right to left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left
,	comma operator	left to right

12. Explain Type Conversion and type casting in C.

> **1. Type Casting:**
> In typing casting, a data type is converted into another data type by the programmer using the casting operator during the program design. In typing casting, the destination data type may be smaller than the source data type when converting the data type to another data type, that's why it is also called narrowing conversion.
> **2. Type conversion :**
> In type conversion, a data type is automatically converted into another data type by a compiler at the compiler time. In type conversion, the destination data type cannot be smaller than the source data type, that's why it is also called widening conversion. One more important thing is that it can only be applied to compatible data types.

13. Find the output of the following code.

```
#include <stdio.h>
void main() {
int k = 8;
int m = 7;
k < m ? k++ : m = k;
printf("%d", k);
}
```

> The code here will throw back an error, as in Line 5, the assignment operator is not declared properly for the relation m=k. If the correction is made as "m==k", then the correct output will be 8.

14. Evaluate the following expressions:

```
1. a+=b*=c-=5 where a=3, b=5, c=8
2. int a,b; float x; a=4; b=5; x=b/a ;
3. int a,b; float x; a=4; b=5; x=(float)b/a;
```

```
#include <stdio.h>
void main() {
int a=3,b=5,c=8;
printf("%d", a+=b*=c-=5);
}
//Output: 18
```

```
#include <stdio.h>
void main() {
int a=4,b=5;
float x=b/a;
printf("%f", x);
}
//Output: 1.000000
```

```
#include <stdio.h>
void main() {
int a=4,b=5;
float x;x=(float)b/a;
printf("%f", x);
}
//Output: 1.250000
```

15. Find the output of the following code.

```
#include <stdio.h>
void main()
{
int a=10,b=2,x=0;
x=a+b*a+10/2*a;
printf("value is =%d",x);
}
//Ouput: value is =80
```

16. Find the output of the following code.

```
#include <stdio.h>
void main()
{
int a = 5 * 3 % 6 - 8 + 3;
printf("%d", a);
}
//Output: -2
```

17. Find the output of the following code.

```
#include <stdio.h>
void main()
{
char a = 'A';
char b = 'B';
int c = a + b % 3 - 3 * 2;
printf("%d\n", c);
}
//Output: 59
```

18. Find the output of the following code.

```
#include <stdio.h>
int main()
{
int a=0;
a = 10 + 5 * 2 * 8 / 2 + 4;
printf("%d", a);
return 0;
}
//Output: 54
```

19. Evaluate the following expressions:

```
1. x = a-b/3 + c*2 - 1 when a = 9, b= 12 & c = 13
2. 10 != 10 || 5 < 4 && 8
3. Evaluate the z=5%3/8*3+4
```

```
#include <stdio.h>
int main()
{
int a=9,b=12,c=13,x;
x = a-b/3 + c*2 - 1;
printf("%d", x);
}
//Output: 30
```

```
#include <stdio.h>
int main()
{
int x;
x = 10 != 10 || 5 < 4 && 8;
printf("%d", x);
}
//Output: 0
```

```
#include <stdio.h>
int main()
{
int z;
z=5%3/8*3+4;
printf("%d", z);
}
//Output: 4
```

20. Evaluate the following expression

```
6*2/ (2+1 * 2/3 + 6) + 8 * (8/4)
```

```
#include <stdio.h>
int main()
{
int x;
x=6*2/ (2+1 * 2/3 + 6) + 8 * (8/4);
printf("%d", x);
}
//Output: 17
```

C-PROGRAMMING QB SOLUTIONS

Module-1

PART-C

@Suhruth

1. Find the output of the following program.

```
#include <stdio.h>
 int main()
 {
 int x = 2, y = 0;
 int z = (y++) ? y == 1 && x : 0;
 printf("%d\n", z);
 return 0;
 }
```

Output: 0

2. Find the output of the following program.

```
#include <stdio.h>
void main()
{
int a, b = 10;
a = -b--;
printf("a = %d, b = %d", a, b);
}
```

Output: a = -10, b = 9

3. Find the output of the following program.

```
#include <stdio.h>
void main()
{
int a, b = 10;
a = b---;
printf("a = %d, b = %d", a, b);
return 0;
}
```

Output: Compilation error.

```
error: expected expression before ‘;’ token
 a = b---;
               ^
warning: ‘return’ with a value, in function returning void
 return 0;
               ^
```

4. Find the output of the following program.

```
#include <stdio.h>
int main()
{
 int a = 2;
 int b = 0;
 int y = (b == 0)? a :(a > b) ? (b = 1): a;
 printf("%d\n", y);
 return 0;
}
```

Output: 2

5. Find the output of the following program.

```
#include <stdio.h>
void main()
{
int a=2, b=0;
int y= (a==0)? a : (a>b)?(b=1):a;
printf(“%d”, y);
}
```

Output: 1

6. Find the output of the following program.

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

```
void main( )
{
 printf(“%d”, -11%5);
}
```

Output: -1

7. Find the output of the following Program.

```
#include <stdio.h>
void main()
{
int x=2, y=0;
int z= (y++) ? y==1 &&x : 0;
printf(“z=%d”, z);
}
```

Output: z=0

8.Find the output of the following Program.

```
#include <stdio.h>
void main()
{
int a=8, b=7;
int c= a<b?a++:b++;
printf(“a=%d, b=%d, c=%d”,a,b,c);
}
```

Output: a=8, b=8, c=7

9.Find the output of the following Program.

```
#include <stdio.h>
void main()
{
int a=8, b=7;
int c = a< b?a =b: ++b;
printf(“a=%d, b=%d, c=%d”,a,b,c);
}
```

Output: a=8, b=8, c=8

10.Find the output of the following Program.

```
#include <stdio h>

void main ( )

{

double x=28;

int y;

y= x%5;

printf (“\n y=%d”, y);

}
```

Output: Compilation error

```
error: invalid operands to binary % (have 'double' and 'int')

y= x % 5;
```