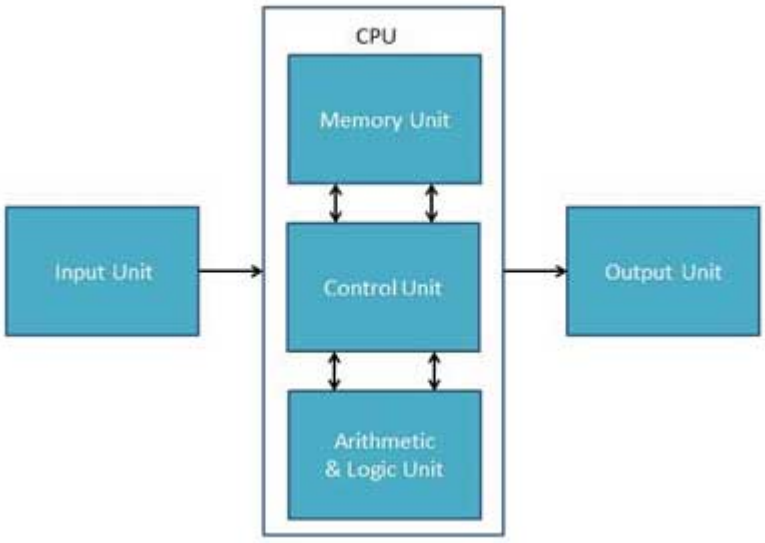
Q.2 a. Draw a block diagram of a computer and explain the functions performed by each block.

Answer:
(Total 10 marks , 1 mark each for input and output unit and 2 marks each for CPU , Memory unit , Control Unit and Arithmetic & Logical Unit)



Input Unit
This unit contains devices with the help of which we enter data into computer. This unit makes link between user and computer. The input devices translate the human being information into the form understandable by computer.

Output Unit
Output unit consists of devices with the help of which we get the information from computer. This unit is a link between computer and users. Output devices translate the computer's output into the form understandable by users.

CPU (Central Processing Unit)
CPU is considered as the brain of the computer. CPU perform all types of data processing operations. It stores data, intermediate results and instructions (program). It controls the operation of all parts of computer.
CPU itself has following three components:

- ALU (Arithmetic Logic Unit)
- Memory Unit
- Control Unit

Control Unit
This unit controls the operations of all parts of computer. It does not carry out any actual data processing operations.
Functions of this unit are:

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It manages and coordinates all the units of the computer.
- It obtains the instructions from the memory, interprets them and directs the operation of the computer.
- It communicates with Input/Output devices for transfer of data or results from storage.
- It does not process or store data.

ALU (Arithmetic Logic Unit)
This unit consists of two subsections namely:

- Arithmetic section
- Logic Section

Arithmetic section
Function of Arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication and division. All complex operations are done by making repetitive use of above operations.
Logic Section
Function of logic section is to perform logic operations such as comparing, selecting, matching and merging of data.

Memory or Storage Unit:
This unit can store instructions, data and intermediate results. This unit supplies information to the other units of the computer when needed. It is also known as internal storage unit or main memory or primary storage or Random access memory (RAM). Its size affects speed, power and capability. There are two types of memories in the computer: Primary memory and secondary memory. Functions of

Memory Unit are:

- It stores all the data to be processed and the instructions required for processing.
- It stores intermediate results of processing.
- It stores final results of processing before these results are released to an output device.
- All inputs and outputs are transmitted through main memory.

b. Briefly explain Impact and Non-Impact Printers along with their characteristics giving two examples of each.

Answer:
Printer is the most important output device, which is used to print information on paper. There are two types of printers:

- Impact Printers
- Non-Impact Printers

Impact Printers
The printers that print the characters by striking against the ribbon and onto the paper, are

called impact printers. Eg. Dot matrix printer , Daisy wheel printer.
Characteristics of Impact Printers are the following:

- Very low consumable costs
- Impact printers are very noisy
- Useful for bulk printing due to low cost
- There is physical contact with the paper to produce an image

Non-impact Printers
The printers that print the characters without striking against the ribbon and onto the paper are called Non-impact Printers. These printers print a complete page at a time, also called as Page Printers. Eg. Laser printer , Ink Jet printer.
Characteristics of Non-impact Printers:

- Faster than impact printers.
- They are not noisy.
- High quality.
- Support many fonts and different character size.

Q.3 a. Briefly explain Static RAM and Dynamic RAM along with their advantages and disadvantages.

Answer:
(Total 6 marks , 3 marks each for Static RAM and Dynamic RAM)
Static RAM (SRAM)
The word **static** indicates that the memory retains its contents as long as power remains applied. However, data is lost when the power gets down due to volatile nature. SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not have to be refreshed on a regular basis.

Because of the extra space in the matrix, SRAM uses more chips than DRAM for the same amount of storage space, thus making the manufacturing costs higher.

Static RAM is used as cache memory needs to be very fast and small.
Characteristics of the Static RAM:

- It has long data lifetime
- There is no need to refresh
- Faster
- Used as cache memory
- Large size
- Expensive
- High power consumption

Dynamic RAM (DRAM)
DRAM, unlike SRAM, must be continually **refreshed** in order for it to maintain the data. This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second. DRAM is used for most system memory because it is cheap and small. All DRAMs are made up of memory cells. These cells are composed of one capacitor and one transistor.

Characteristics of the Dynamic RAM:

- It has short data lifetime
- Need to refresh continuously
- Slower as compared to SRAM
- Used as RAM
- Lesser in size
- Less expensive
- Less power consumption

b. Compare and contrast a single user with multi-user operating system. Use suitable examples.

Answer:

c. Convert the decimal number 2536 into octal format.

Answer:

Successive Dividers	**Original Numbers and partial quotients**	**Remainders**
8	2536	0 (LSB)
8	317	5
8	39	7
8	4	4 (MSB)
	0	

$(2536)_{10} = (4750)_{8}$

Hence the octal equivalent of decimal number 2536 is 4750.

Q.4 a. What is a complete directive in C programming language? Name and briefly explain any one such directive.

Answer:

(Any one directive such as # define or #include can be explained, 1 more for explaining compiler directive)

b. Describe the basic structure of a C program.

Answer:

Structure of C program is defined by set of rules called protocol, to be followed by programmer while writing C program. All C programs are having sections/parts which are mentioned below.

1. Documentation section
2. Link Section
3. Definition Section
4. Global declaration section
5. Function prototype declaration section
6. Main function

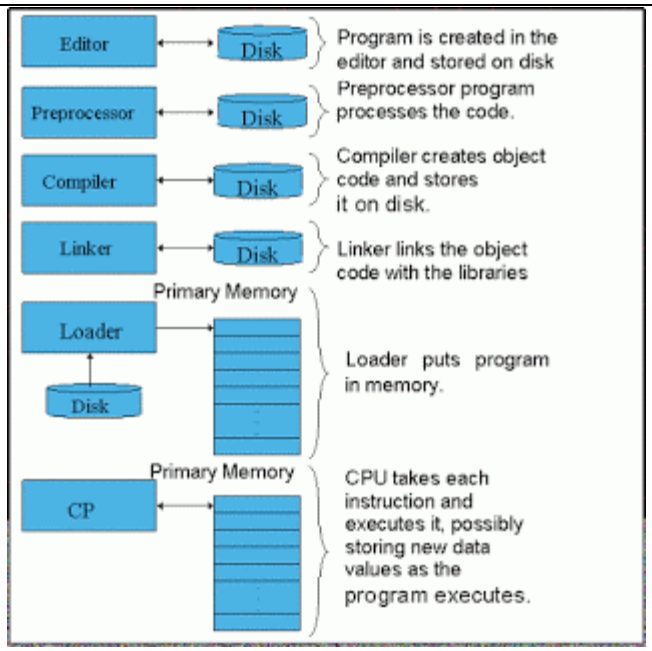
7. User defined function definition section

Please note that a C program may not have all below mentioned sections except main function and link sections.

S.No	Sections	Description
1	Documentation section	We can give comments about the program, creation or modified date, author name etc in this section. The characters or words or anything which are given between "/*" and "*/", won't be considered by C compiler for compilation process.These will be ignored by C compiler during compilation. Example : /* comment line1 comment line2 comment 3 */
2	Link Section	Header files that are required to execute a C program are included in this section
3	Definition Section	In this section, variables are defined and values are set to these variables.
4	Global declaration section	Global variables are defined in this section. When a variable is to be used throughout the program, can be defined in this section.
5	Function prototype declaration section	Function prototype gives many information about a function like return type, parameter names used inside the function.
6	Main function	Every C program is started from main function and this function contains two major sections called declaration section and executable section.
7	User defined function section	User can define their own functions in this section which perform particular task as per the user requirement.

c. List the steps in writing and executing a C program.

Answer:
(Total six marks, and 2 marks for the diagram)



1. Program is first written in any text editor or using any IDE like Turbo C , Borland C etc.
2. The program is then saved with an extension of .c
3. The program is then compiled. If there are some syntax errors they are listed. These errors are removed and then the program is recompiled.
4. Once the program becomes error free an object code is created with an extension of .obj.
5. The linker then links this object code with the function libraries and then an executable file is created with an extension of .exe.
6. The loader then loads this executable file in the memory and the program is executed.

Q.5 a. What do you mean by storage classes? Briefly explain the various storage classes available in C.

Answer:

Storage classes

Storage class specifiers in C language tells the compiler where to store a variable, how to store the variable, what is the initial value of the variable and life time of the variable.

Syntax: storage_specifier data_type variable _name

Types of Storage Class Specifiers in C:

There are 4 storage class specifiers available in C language. They are,

1. auto
2. extern
3. static
4. register

S.No.	Storage Specifier	Storage place	Initial / default	Scope	Life

			value		
1	**Auto**	CPU Memory	Garbage value	local	Within the function only.
2	**Extern**	CPU memory	Zero	Global	Till the end of the main program. Variable definition might be anywhere in the C program
3	**Static**	CPU memory	Zero	local	Retains the value of the variable between different function calls.
4	**Register**	Register memory	Garbage value	local	Within the function

b. Identify the function used to accept formatted input in C programming language. Specify its syntax. Also specify the form in which it can be used to accept
(i) Integer Number
(ii) Real Number and (iii) Character String **(4)**

Answer:

c. What are symbolic constants in C? Explain with its syntax and use.

Answer:

(Total 4 marks, 3 marks for explanation and 1 mark for its syntax.)

Symbolic constant in c Language

A symbolic constant is name that substitute for a sequence of character that cannot be changed. The character may represent a numeric constant, a character constant, or a string. When the program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence. They are usually defined at the beginning of the program. The symbolic constants may then appear later in the program in place of the numeric constants, character constants, etc., that the symbolic constants represent.

For example

A **C program** consists of the following symbolic constant definitions.

```
#define PI 3.141593
#define TRUE 1
#define FALSE 0
```

#define PI 3.141593 defines a symbolic constant PI whose value is 3.141593. When the program is preprocessed, all occurrences of the symbolic constant PI are replaced with the replacement text 3.141593.

Q.6 a. What is the output of the following code? **(6)**

```
main( )
{
   float a,b,c,x,y,z;

   a=9;
   b=12;
   c=3;

   x=a–b/3+c*2–1;
   y=a–b/(3+c)*(2–1);
   z=a–(b/(3+c)*2)–1;
   printf(“x=%f\n”, x);
   printf(“y=%f\n”, y);
   printf(“z=%f\n”, z);
}
```

Answer: Output of the code is

x = 10.000000
y = 7.000000
z = 4.000000

b. Briefly explain the various special operators available in C

Answer:

Special Operators in C:

Below are some of special operators that C language offers.

S.no	Operators	Description
1	&	This is used to get the address of the variable. Example : &a will give address of a.
2	*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
3	Sizeof ()	This gives the size of the variable. Example : size of (char) will give us 1.

c. Briefly explain any four mathematical functions in C along with their syntax.

Answer:

(Total 4 marks , 1 mark for each function explained with correct syntax.)

C has a wide variety of mathematical functions. The corresponding definitions and declarations are in math.h. Following are some of the mathematical functions in C .

1. pow( )

```
#include <math.h>
double pow(double x, double y);
```

// calculate *x* raised to the exponent *y*. On success, pow return the value calculated.

2. Sqrt)

```
#include <math.h>
double sqrt(double x);
```

```
//sqrt computes the positive square root of the argument. On success, the
   //   square root is returned.
```

3. Round()

```
#include <math.h>
double round(double x);
//The round functions round their argument to the nearest integer
//value in floating-point format and x rounded to an integral value.
```

4. sin, cos - sine or cosine

```
#include <math.h>
double sin(double x);
double cos(double x);
//sin and cos compute (respectively) the sine and cosine of the
//argument x. Angles are specified in radians. The sine or cosine
//of x is returned.
```

Q.7 a. Differentiate between the following:
(i) while & do.. while loop in C
(ii) while loop and for loop in C
(iii) break & continue statement in C

Answer:

(i)

while loop

- It is an entry controlled loop
- Here the condition is tested in the beginning
- The loop may run zero or more times i.e. when the condition evaluated to false in the beginning itself the loop is not executed at all.

do.... while loop

- It is an exit controlled loop
- Here the condition is tested in the end
- The loop will run one or more times i.e. the loop is executed once and then the condition evaluated. So this will always run atleast for once.

(ii)

while loop and for loop

For loop runs for a fixed number of times and hence is usually used when we know in advance as to how many times the loop is to be executed.
While is an entry controlled loop and runs zero or more times depending on the condition. This loop is therefore used usually when we do not know in advance as to how many times the loop has to be executed .

(iii)

break statement in C:

- Break statement is used to terminate the while loops, switch case loops and for loops from the subsequent execution.
- The break command will exit the most immediately surrounding loop regardless

of what the conditions of the loop are.

- Break is useful if we want to exit a loop under special circumstances.
- Syntax: break;

continue statement in C:

- Continue statement is used to continue the next iteration of for loop, while loop and do-while loops. So, the remaining statements are skipped within the loop for that particular iteration.
- If you are executing a loop and hit a continue statement, the loop will stop its current iteration, update itself (in the case of for loops) and begin to execute again from the top.
- Essentially, the continue statement is saying "this iteration of the loop is done, let's continue with the loop without executing whatever code comes after me."
- Syntax : continue;

b. Write a C program to display all prime numbers between Two Numbers entered by user. (8)

Answer:

```
/* C program to display all prime numbers between Two Numbers entered by user. */
#include <stdio.h>
int main()
{
  int n1, n2, i, j, flag;
  printf("Enter two numbers(intervals): ");
  scanf("%d %d", &n1, &n2);
  printf("Prime numbers between %d and %d are: ", n1, n2);
  for(i=n1+1; i<n2; ++i)
  {
     flag=0;
     for(j=2; j<=i/2; ++j)
     {
       if(i%j==0)
       {
         flag=1;
         break;
       }
     }
     if(flag==0)
       printf("%d ",i);
  }
  return 0;
}
```

c. Rewrite the following function y using conditional operator:

y = 1.5x + 3 for x≤ 2
y = 2x + 5 for x > 2

Answer:

y = (x>2) ? (2*x+5) : (1.5*x+3);

Q.8 a. Define Recursion. Write a program to find the factorial of a number using recursion.

Answer:

(Total 6 marks, 1 mark for defining recursion 1 mark for syntax and 4 marks for program)

A function calling itself is known as recursion. This technique is used to solve certain types of problems which are recursive / repetitive in nature like finding a raised to the power b , factorial of a number etc.

Ex.

```
#include <iostream.h>
void main()
{
-----
f();
-----
}
void f()
{
  f();
-----
}
```

Here the function f() calls f() itself.

```
#include <iostream.h>
#include <conio.h>
long int fact( long int);
void main( )
{
 long int f,n;
 cout << "Enter a number:";
 cin>> n;
 f = fact( n );
 cout << "The factorial of the number is:" << f<< endl;
getch();
}

long int fact( long int)
{
long int value = 1;
if ( n== 1) return (value);
```

```
else
 {
  value = n * fact (n – 1 );
  return(value);
  }
```

b. Distinguish between the following:
(i) Actual parameter and Formal parameter
(ii) Local variable and Global variable **(8)**

Answer:

(i) **Actual Parameter and Formal Parameter**

If a function is to use arguments, it must declare variables that accept the values of the arguments passed to it by the calling function. These variables are called the **formal parameters** of the function. The formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit. **Actual Parameters** on the other hand are those parameters which contain the values which are sent to the function , on which the function executes. Values of Actual parameters are sent to the function and stored there locally in formal parameters.

(ii) Local variable and Global Variable

Local variables

- The scope of local variables will be within the function only.
- These variables are declared within the function and can't be accessed outside the function.

Global Variables

- The scope of global variables will be throughout the program. These variables can be accessed from anywhere in the program.
- This variable is defined outside the main function. So that, this variable is visible to main function and all other sub functions.
- The global variables will hold their value throughout the lifetime of your program

Q.9 a. Write a program to find the Maximum and Minimum marks obtained by students in a class of 60 student.

Answer:

```
/* program to find the Maximum and Minimum marks obtained in a class of 60 students
*/
#include<stdio.h>
#include<conio.h>
void main()
{
 int marks[60] , i , max, min;
 /* read marks obtained by 60 students in an array */
 for( i = 0 ; i <60 ; i++ )
  scanf("%d",&marks[i]);
```

```
   max = marks[0];
   min = marks[0];

  for( i = 0 ; i < 60 ; i++ )
  {
   if ( marks[i] > max )
    max = marks[i];
    }

for( i = 0 ; i < 60 ; i++ )
  {
   if ( marks[i] < min )
    min = marks[i];
   }
  /* print the Result */
  printf("\n The Highest Marks scored are : %d" , max);
 printf("\n The Lowest Marks scored are : %d" , min);
  getch();
}
```

b. Briefly explain the following string functions in C. Use suitable examples.

(i) strcat( )
(ii) strcpy( )
(iii) strlen( )
(iv) strcmp( )

Answer:

strcat( )

strcat( ) function in C language concatenates two given strings. It concatenates source string at the end of destination string. Syntax for strcat( ) function is given below.

```
char * strcat ( char * destination, const char * source );
```

Example :

```
strcat ( str2, str1 );   /* str1 is concatenated at the end of str2. */
strcat ( str1, str2 );   /* str2 is concatenated at the end of str1.  */
```

strcpy( )

strcpy( ) function copies contents of one string into another string. Syntax for strcpy function is given below.

```
char * strcpy ( char * destination, const char * source );
```

Example:

```
strcpy ( str1, str2)   /* It copies contents of str2 into str1.  */
strcpy ( str2, str1)   /* It copies contents of str1 into str2.  */
```

- If destination string length is less than source string, entire source string value won't be copied into destination string.
- For example, consider destination string length is 20 and source string length is

30. Then, only 20 characters from source string will be copied into destination string and remaining 10 characters won't be copied and will be truncated.

strlen( )

strlen( ) function in C gives the length of the given string. Syntax for strlen( ) function is given below.

size_t strlen (const char * str);

- strlen( ) function counts the number of characters in a given string and returns the integer value.
- It stops counting the character when null character is found. Because, null character indicates the end of the string in C.

strcmp( )

strcmp( ) function in C compares two given strings and returns zero if they are same.

- If string1 < string2, it returns < 0 value. If string1 > string2, it returns > 0 value. Syntax for strcmp( ) function is given below.
  int strcmp (const char * str1, const char * str2);

strcmp( ) function is case sensitive. i.e, "A" and "a" are treated as different characters.

TEXT BOOK

I. Computer Concepts and Programming in C, E. Balagurusamy, Tata McGraw-Hill, 2010