POP Model paper – 2

1. **a) Explain the structure of C program in detail. Write a sample program to demonstrate the components in the structure of C program.**

Ans:

The general structure of a C program contains following sections:

1. Documentation section: This section consists of comment lines which include the name of the program, the name of the programmer, the author and other details like time and date of writing the program. Documentation section helps anyone to get an overview of the program.

Example: 1)Multiline comments(/*------*/) & 2)Singleline comments(//--------)

2. Link section: The link section consists of the header files of the functions that are used in the program. It provides instructions to the compiler to link functions from the system library such as using the #include directive.

Example: #include<stdio.h> , #include<math.h>

3. Definition section: All the symbolic constants are written in the definition section. Macros are known as symbolic constants.

Example: #define PI 3.14 , #define MAXSIZE 10

4. Global declaration: The global variables that can be used anywhere in the program are declared in the global declaration section. This section also declares the user defined functions.

5. Main function: It is necessary to have one main() function section in every C program. This section contains two parts, declaration and executable part. The declaration part declares all the variables that are used in executable part. These two parts must be written in between the opening and closing braces. Each statement in the declaration and executable part must end with a semicolon (;). The execution of the program starts at opening braces and ends at closing braces.

i) Declaration part: The declaration part declares all the variables used in the executable part.

ii) Executable part: There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.

6. User defined functions: The subprogram section contains all the user defined functions that are used to perform a specific task. These user defined functions are called in the main() function. If the program is a multifunction program then the sub program section contains all the user-defined functions that are called in the main () function. User-defined functions are generally placed immediately after the main () function, although they may appear in any order.

b) Demonstrate formatted output of integer in C with suitable example

Ans:

Program to add two numbers:

```
#include<stdio.h>
void main()
{
    int n1,n2,sum;
    printf("Enter two integers\n");
    scanf("%d %d",&n1,&n2);
    sum=n1+n2;
    printf("Sum = %d",sum);
}
```

c) Discuss different types of error occur in program
Ans:
There are 5 different types of errors in C.
1. Syntax error: Syntax errors occur when a programmer makes mistakes in typing the code's syntax correctly. Syntax errors are sometimes also called *compilation errors* because they are always detected by the compiler.
2. Runtime error: Errors that occur during the execution (or running) of a program are called RunTime Errors. These errors occur after the program has been compiled successfully. When a program is running, and it is not able to perform any particular operation, it means that we have encountered a run time error.
Runtime errors can occur because of various reasons. Some of the reasons are:
i) Mistakes in the code
ii) Memory leaks
iii) Mathematically incorrect operations
iv) Undefined variables.
3. Logical error: Sometimes, we do not get the output we expected after the compilation and execution of a program. Even though the code seems error free, the output generated is different from the expected one. These types of errors are called Logical Errors.
4. Semantic error: Errors that occur because the compiler is unable to understand the written code are called Semantic Errors. A semantic error will be generated if the code makes no sense to the compiler, even though it is syntactically correct.
5. Linker error: Linker errors are the errors encountered when the executable file of the code can not be generated even though the code gets compiled successfully. This error is generated when a different object file is unable to link with the main object file. We can run into a linked error if we have imported an incorrect header file in the code, we have a wrong function declaration, etc.

2. **a) Explain the various rules for forming identifiers names. Give examples for valid and invalid identifiers for the same.**
Ans:
Variable is a named location in a memory where a program can manipulate the data.
The rules to construct variables are:
- They can contain letters, digits and underscores.
- They should not begin with any special characters except underscore (_).
- They are case sensitive.
- They cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words cannot be used as variables(such as int, float, char etc.).

Valid: num2, a_2, _apple.
Invalid: $num1, +add, 199_space, #12.

b) Mention various output devices and explain hardcopy devices.
Ans:
The various output devices are:

- Monitor
- Printer
- Speaker
- Plotter
- Projectors etc.

Hard copy devices are the devices which prints the data on a paper.
i)Printer: Printer is the most important output device, which is used to print information on paper. There are two types of printers:

- **Impact printers:** The printers that print the characters by striking against the ribbon and onto the paper are called impact printers. Characteristics of Impact Printers are following:
  - Very low consumable costs.
  - They are very noisy.
  - Useful for bulk printing due to low cost.
  - There is physical contact between paper and printer pins to produce images, characters etc.
  - They are two types:
    - ❖ Character printer
    - ❖ Line printer
- **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. Characteristics of non-impact printers are following:
  - Faster than impact printers.
  - They are not noisy.
  - High quality printing.
  - Supports many fonts and different character size.
  - They are two types:
    - ❖ Laser printer
    - ❖ Inkjet printer

c) Discuss the variants of microcomputer that are widely used today.

Ans:

i. **Desktop Computer:** a personal or micro-mini computer sufficient to fit on a desk.

ii. **Laptop Computer:** a portable computer completes with an integrated screen and keyboard. It is generally smaller in size than a desktop computer and larger than a notebook computer.

iii. **Palmtop Computer/Digital Diary/Notebook/PDAs(Personal Digital Assistant): A** hand-sized computer, Palmtop, does not have keyboard, but its screen serves both as an input and output device.

iv. **Smartphones and Embedded Computers:**

- The relentless drive toward miniaturization has led to the emergence of two types of devices called smartphones and embedded computers.
- The smartphone is a general-purpose computer that is also capable of making phone calls.The smartphone has a powerful processor, usually with multiple cores. It also supports gigabytes of main memory but doesn't have a hard disk for secondary storage. Smartphones today run a well-developed operating system (Android or iOS), and can run a wide range of applications (popularly called "apps"). Like a computer, a smartphone has a keyboard and a high-resolution display, both operated by touch or a stylus. Applications are written in a high-level language.
- The embedded computer is a small computer-like system that is part of a larger system. Embedded computers (also called micro-controllers) arrived before smartphones. These are very small circuits containing a CPU, non-volatile memory, and input and output handling facilities. They are embedded into many of the machines that we use—cars, washing machines, MP3 players and cameras. The processor here runs a single unmodifiable program stored in memory. Embedded computers can't match the capabilities of a smartphone, but they don't need to. Because the processor is dedicated to a specific task, its capabilities are just adequate to operate the device.

3. **a) Demonstrate the functioning of Bitwise operator in C.**

Ans:

There are 6 bitwise operators:

i) one's complement (~) : All 0 changes to 1 and all 1 change to 0.
Example: ~35 = ~00100011 = 11011100 = 220

ii) bitwise AND (&) : The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.
Example: 12 & 25 = 00001100 & 00011001 = 00001000 = 8

iii) bitwise OR (|) : The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1.
Example: 12 | 25 = 00001100 | 00011001 = 00011101 = 29

iv) bitwise XOR (^) : The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite.
Example : 12 ^ 25 = 00001100 ^ 00011001 = 00010101 = 21

v) left shift (<<) : Left shift operator shifts all bits towards left by a certain number of specified bits.
Example : 212>>2 = 11010100>>2 = 00110101

vi) right shift (>>) : Right shift operator shifts all bits towards right by certain number of specified bits.
Example: 212<<2 = 11010100<<2 = 1101010000

b) Write a C program to find roots of quadratic equation.

Ans:

```
#include<stdio.h>
#include<math.h>
void main()
{
float a,b,c,d,real,img,root1,root2;
printf("\nEnter the values of a,b,c\n");
if(a = = 0)
printf("\n Invalid coefficients, roots cannot be computed\n");
else
{
d=b*b-4*a*c;
printf("\n The value of discriminant(d) is = %f",d);
if(d = = 0)
{
root1=root2= -b/(2*a);
printf("\n The real and equal roots are: root1 = root2 = %.2f",root1);
}
else if(d > 0)
{
root1 = (-b+sqrt(d))/(2*a);
root2 = (-b-sqrt(d))/(2*a);
printf("\n The real and distinct roots are: root1=%.2f & root2=%.2f",root1,root2);
}
else
{
real = -b/(2*a);
img = sqrt(fabs(d))/(2*a);
printf("\nThe Complex roots are: root1=%.2f+%.2fi & root2=%.2f-%.2fi",real,img,real,img);
}
}
}
```

c) Distinguish between the break and continue statement.

Ans:

Break Statement	Continue Statement
The Break statement is used to exit from the loop constructs.	The continue statement is not used to exit from the loop constructs.
The break statement is usually used with the switch statement, and it can also use it within the while loop, do-while loop, or the for-loop.	The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.
When a break statement is encountered then the control is exited from the loop construct immediately.	When the continue statement is encountered then the control automatically passed from the beginning of the loop statement.
Break statements uses switch and label statements.	It does not use switch and label statements.
Leftover iterations are not executed after the break statement.	Leftover iterations can be executed even if the continue keyword appears in a loop.
Syntax: break;	**Syntax:** continue;

4. a) Illustrate Nested loops in C with suitable example.

Ans:

Program to read and display the elements of a 2d array.

```
#include<stdio.h>
void main()
{
int M[5][5],i,j;
printf("enter a values for matrix M in 5 rows and 5 columns:");
for(i=0;i<5;i++)
for(j=0;j<5;j++)
scanf("%d",&M[i][j]);
printf("the elements of matrix are: \n");
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
printf("%d\t",M[i][j]);
printf("\n");
}
}
```

b) Write a C program to print whether a given number is palindrome or not.

Ans:

```
#include<stdio.h>
void main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
printf("palindrome number ");
else
printf("not palindrome");
}
```

c) Explain switch statement with syntax. Write a C program to simulate calculator.
Ans:
Switch statement is also called as multi-way branching statement. The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated, with that case is executed.
Syntax:

```
switch(expression)
{
case value-1: block-1;
              break;
case value-2: block-2;
              break;
--------------------------
--------------------------
default: default-block;
              break;
}
```

Example:

```
#include<stdio.h>
void main()
{
char op;
float n1,n2,result=0;
printf("\nEnter an operator(+,-,*,/): ");
scanf("%c",op);
printf("\nEnter the value of two operands: n1 and n2: ");
scanf("%f %f",&n1,&n2);
switch(op)
{
case '+':result = n1+n2;
         break;
case '-':result = n1-n2;
         break;
case '*':result = n1*n2;
         break;
case '/':result = n1/n2;
         break;
default:printf("\nYou have entered an invalid operator");
}
printf("%f %c %f = %f",n1,op,n2,result);
}
```

5. a) Write a C program to implement Bubble sort technique(ascending order).

Ans:

```
#include<stdio.h>
void main( )
{
int a[10],n,i,j,temp;
printf("enter the size of array:\n");
scanf("%d",&n);
printf("enter the array elements:\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(i=0;i<n-1;i++)
for(j=0;j<n-i-1;j++)
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
printf(" the sorted array is:\n");
for(i=0;i<n;i++)
printf("%d",a[i]);
}
```

b) Illustrate the concept of recursive function with example.

Ans:

```
#include<stdio.h>
int fact(int n);
void main()
{
int num;
printf("Enter a number\n");
scanf("%d",&num);
printf("Factorial of %d = %d",num,fact(num));
}
int fact(int n)
{
if(n= = 1)
return 1;
return (n*fact(n-1));
}
```

c) Discuss various scope of variables.

Ans:

A scope in any programming is a region of the program where a defined variable can have its existence and beyond that variable cannot be accessed. There are three places where variables can be declared in C programming language:

- Inside a function or a block which is called **local** variables,
- Outside of all functions which is called **global** variables.
- In the definition of function parameters which is called **formal** parameters.
- There are 2 scopes:

i) Local variable scope: Variables that are declared inside a function or block are called local variables. They can be used only by statements that are inside that function or block of code.

Example:

```
#include <stdio.h>
void main ()
{
int a,b,c;
a = 10;
b = 20;
c = a + b;
printf ("value of a = %d, b = %d and Sum = %d\n", a, b, c);
}
```

ii) Global variables scope: Global variables are defined outside of a function, usually on top of the program. The global variables will hold their value throughout the lifetime of your program and they can be accessed inside any of the functions defined for the program. A global **variable** can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration.

Example:

```
#include <stdio.h>
int g;
void main()
{
int a, b;
a = 10;
b = 20;
g = a + b;
printf ("value of a = %d, b = %d and Sum = %d\n", a, b, g);
}
```

6. a) Differentiate between call by value and call by reference. Using suitable example.

Ans:

Call by value	Call by reference
A copy of the value is passed into the function	An address of value is passed into the function
Changes made inside the function is limited to the function only. The values of the actual parameters do not change by changing the formal parameters.	Changes made inside the function validate outside of the function also. The values of the actual parameters do change by changing the formal parameters.
Actual and formal arguments are created at the different memory location	Actual and formal arguments are created at the same memory location
Example:	Example:
#include<stdio.h>	#include<stdio.h>
void swapping(int a, int b);	void swapping(int *a, int *b);
void main()	void main()
{	{
int x,y;	int x,y;
printf("Enter x, y values\n");	printf("Enter x, y values\n");
scanf("%d %d",&x,&y);	scanf("%d %d",&x,&y);
printf("Before swapping\n");	printf("Before swapping\n");
printf("%d %d",x,y);	printf("%d %d",x,y);
swapping(x,y);	swapping(&x,&y);
}	}
void swapping(int a, int b)	void swapping(int *a, int *b)
{	{
int temp;	int temp;
temp = a;	temp = *a;
a = b;	*a = *b;
b = temp;	*b = temp;
printf("After swapping\n");	printf("After swapping\n");
printf("%d %d",a,b);	printf("%d %d",*a,*b);
}	}

b) Write a C program to transpose a MxN matrix.

Ans:

```
#include<stdio.h>
void main()
{
int M[10][10], T[10][10],i,j,m,n;
printf("enter the row size of array:");
scanf("%d",&m);
printf("enter the column size of array:");
scanf("%d",&n);
printf("enter a values for matrix M:");
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d",&M[i][j]);
for(i=0;i<m;i++)
for(j=0;j<n;j++)
T[i][j]=M[j][i];
printf("the elements of transpose matrix T are: \n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
printf("%d\t",T[i][j]);
printf("\n");
}
}
```

c) Discuss the various storage classes.
<u>Ans:</u>

- A storage class defines the scope (visibility) and life-time of variables and/or functions within a C Program. These specifiers precede the type that they modify.
- There are the following storage classes, which can be used in a C Program:
  - auto
  - register
  - static
  - extern

Storage classes	Storage place	Default value	Scope	Lifetime
auto	RAM	Garbage Value	Local	Within the function
register	Register	Garbage Value	Local	Within the function
static	RAM	Zero	Local	Till the end of the main program, Retains value between multiple functions call
extern	RAM	Zero	Global	Till the end of the main program, Maybe declared anywhere in the program

7. a) Mention various operations that can be performed on string using built-in functions. Explain any two function.
<u>Ans:</u>
i) strlen( )
ii) strupr( )
iii) strlwr( )
iv) strcmp( )
v) strcat( )
vi) strcpy( )
vii) strrev( )
viii) strstr( )
ix) strchr( )
x) strrchr( )

<u>1)strlen:</u> This string function is basically used for the purpose of computing the length of string.
Syntax: strlen(str);
Example:

```
#include<stdio.h>
void main()
{
char str[10]= “COMPUTER”;
int len = strlen(str);
printf(“The length of string is = %d”,len);
}
```

<u>2)strupr:</u> This string function is used to convert the string to uppercase.
Syntax: strupr(str);
Example:

```
#include<stdio.h>
void main()
{
char str[10]= “computer”;
printf(“The uppercase of string is = %s”,strupr(str));
}
```

b) Develop a program using pointer to compute the sum, mean and standard deviation of all element stored in array of N real number.

<u>Ans:</u>

```
#include<stdio.h>
#include<math.h>
int main()
{
float a[10], *ptr, mean, std, sum=0, sumstd=0;
int n,i;
printf("\n Enter the no of elements\n");
scanf("%d",&n);
printf("\n Enter the array elements\n");
for(i=0;i<n;i++)
scanf("%f",&a[i]);
ptr=a;
for(i=0;i<n;i++)
{
sum=sum+ *ptr;
ptr++;
}
mean=sum/n;
ptr=a;
for(i=0;i<n;i++)
{
sumstd=sumstd + pow((*ptr - mean),2);
ptr++;
}
std= sqrt(sumstd/n);
printf("\n Sum=%.3f\t",sum);
printf("\n Mean=%.3f\t",mean);
printf("\n Standard deviation=%.3f\t",std);
return 0;
}
```

c) Explain how strings are represented in main memory.

<u>Ans:</u>

<u>Representing strings in C:</u>

- Stored in arrays of characters
- Array can be of any length
- End of string is indicated by a delimiter character, the zero character – '\0'(null character)
- Character array is one-dimensional array of characters and is also called as String.
- Example : - char name[10] = " Johnny " ;
- The name[10] is a character array which stores the string **JOHNNY** as sequence of characters as shown below with null terminated character(\0).

J	O	H	N	N	Y	\0
name[0]	name[1]	name[2]	name[3]	name[4]	name[5]	name[6]

8. a) Write a program to compare two strings without using built-in function.

Ans:

```
#include <stdio.h>
int compare(char[],char[]);
void main()
{
char str1[20];
char str2[20];
printf("Enter the first string : ");
scanf("%s",str1);
printf("Enter the second string : ");
scanf("%s",str2);
int c= compare(str1,str2);
if(c==0)
printf("strings are same");
else
printf("strings are not same");
return 0;
}
int compare(char a[],char b[])
{
int flag=0,i=0;  // integer variables declaration
while(a[i]!='\0' && b[i]!='\0')
{
if(a[i]!=b[i])
{
flag=1;
break;
}
i++;
}
if(flag==0)
return 0;
else
return 1;
}
```

b) What is pointer? Discuss pointer arithmetic with suitable C code.

Ans:

Pointer is a special variable which stores the address of another variable.

Program to implement simple calculator using pointers:

```
#include<stdio.h>
void main()
{
char op;
float n1,n2,result=0,*p1,*p2;
printf("\nEnter an operator(+,-,*,/): ");
scanf("%c",op);
printf("\nEnter the value of two operands: n1 and n2: ");
scanf("%f %f",&n1,&n2);
p1=&n1;
p2=&n2;
switch(op)
{
case '+':result = *p1 + *p2;
break;
case '-':result = *p1 - *p2;
break;
case '*':result = *p1* *p2;
break;
case '/':result = *p1/*p2;
break;
default:printf("\nYou have entered an invalid operator");
}
printf("%f %c %f = %f",*p1,op,*p2,result);
}
```

c) Explain gets()and puts() function with example.
Ans:

gets():
gets() is used to read string from the standard input device until newline character not found, use of gets() may risky because it does not check the array bound.
Syntax: gets(str);
Example:
```
#include<stdio.h>
#include<string.h>
void main()
{
 char  name[20];
gets(name);
printf("\nYou Entered: %s",name);
}
```

puts():
puts() is used to write(display) string on standard output device until newline character not found. It terminates the line with a newline character. It returns an EOF(-1) if an error occurs and returns a positive number on success.
Syntax: puts(str);
Example:
```
#include<stdio.h>
#include<string.h>
void main()
{
char name[50];
printf("Enter your name\n");
scanf("%s",name);
printf("Your name is : ");
puts(name);
}
```

9. **a) Explain various modes in which file can be opened for processing.**
Ans:

Mode	Description
r	Opens a file in read mode
w	Opens or creates a text file in write mode
a	Opens a file in append mode
r+	Opens a file in both read and write mode
w+	Opens a file in both read and write mode
a+	Opens a file in both read and write mode

b) Implement structure to read, write and compute average marks of the students. List the students scoring above and below the average marks for a class of n students.

Ans:

```
#include<stdio.h>
struct student
{
int marks;
}st[10];
void main()
{
int i,n;
float total=0,avgmarks;
printf("\nEnter the number of students in class(<=10):");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter student %d marks :",i+1);
scanf("%d",&st[i].marks);
}
for(i=0;i<n;i++)
total = total + st[i].marks;
avgmarks=total/n;
printf("\nAverage marks = %.2f",avgmarks);
for(i=0;i<n;i++)
{
if(st[i].marks>=avgmarks)
printf("\n student %d marks = %d above average",i+1,st[i].marks);
else
printf("\n student %d marks = %d below average",i+1,st[i].marks);
}
}
```

c) What are enumeration variable? How are they declared.

Ans:

- An enumeration provides the data type with a set of values. An enumeration constant is a type of an integer. A variable type takes and stores the values of the enumeration set defined by that type. In C programming, an enumeration type (also called enum) is a data type that is used to assign integral constants to set of names.
- They are declared by using keyword enum in place of datatye.
- **Declaration: Syntax→ enum enum_variable_name{ele1,ele2,ele3,…….,elen};**
- **Example:**

```
#include <stdio.h>
enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
void main()
{
enum week today;
today = Wednesday;
printf("Day %d",today+1);
}
```

10. a) Write a short note on functions used to Read data from a file, Write data to a file.

Ans:

C provides the following set of functions to read data from a file.

- **fscanf():** It is used to read formatted data from the stream.
  Syntax: int fscanf(FILE *stream, const char *format, …);

- **fgets():** It is used to get a string from a stream.
  Syntax: char *fgets(char *str, int n, FILE *stream);

- **fgetc():** It returns the next character from stream, and EOF if the end of file is reached, or if there is an error.
  Syntax: int fgetc(FILE *stream);

- **fread():** It is used to read data from a file.
  Syntax: int fread(void *str, size_t size, size_t num, FILE *stream);

C provides the following set of functions to write data into a file.

- **fprintf():** It is used to write formatted output to stream.
  Syntax: int fprintf(FILE *stream, const char *format, …);

- **fputs():** It is used to write a line to a file.
  Syntax: int fputs(const char *str, FILE *stream);

- **fputc():** It is used to write a character to the stream.
  Syntax: int fputc(int c, FILE *stream);

- **fwrite():** It is used to write data to a file.
  Syntax: int fwrite(const void *str, size_t size, size_t count, FILE *stream);

b) Differentiate structures and unions with syntax and example.

Ans:

Si. No.	Sturcture	Union
1	**Structures** is a user-defined data type available in C that allows to combining of data items of different kinds.	**Union** is a special data type available in C that allows storing different data types in the same memory location.
2	It allocates memory equal to sum of memory allocated to its each individual member.	It allocates piece of memory that is Large enough to hold the Largest variable of type in union.
3	Each member have their own memory space	One block is used by all the members of union
4	Structure can not be implemented in shared memory	Union is the Best environment where memory is shared.
5	It has less Ambiguity	As memory is shared, Ambiguity is more in union.
6	Self referential structure can be implemented in data structure.	Self referential union can not be implemented.
7	All members of structure can be accessed at a time	Only one member is accessed at a time.
8	Size of structure is equal to the sum of the size of member variables.	Size of union is equal to the size of the large sized member variable.

9	**Syntax:**
	struct [structure name]
	{
	member definition;
	member definition;
	...
	member definition;
	};
		Syntax:
		union [union name]
		{
		member definition;
		member definition;
		...
		member definition;
		};

10	Example structure declaration:
	struct sVALUE
	{
	int ival;
	float fval;
	char *ptr;
	}s;
	Here size of struct is 8 Bytes.
		Example union declaration:
		union uVALUE
		{
		int ival;
		float fval;
		char *ptr;
		}u;
		Here size of union is 4 Bytes.

c) How to detect END-OF-FILE.

Ans:

The End of the File (EOF) indicates the end of input. In C, getc() returns EOF when end of file is reached. getc() also returns EOF when it fails. So, only comparing the value returned by getc() with EOF is not sufficient to check for actual end of file. To solve this problem, C provides feof() which returns non-zero value only if end of file has reached, otherwise it returns 0. For example, consider the following C program to print contents of file *test.txt* on screen. In the program, returned value of getc() is compared with EOF first, then there is another check using feof(). By putting this check, we make sure that the program prints *"End of file reached"* only if end of file is reached. And if getc() returns EOF due to any other reason, then the program prints *"Something went wrong"*.

Example:

```
#include <stdio.h>
 int main()
{
  FILE *fp;
  fp = fopen("test.txt", "r");
  int ch = getc(fp);
  while (ch != EOF)
  {
    putchar(ch);
    ch = getc(fp);
  }
  if (feof(fp))
  printf("\n End of file reached.");
  else
  printf("\n Something went wrong.");
  fclose(fp);
 getchar();
 return 0;
}
```