Q2. (a) Explain the working of computer with stored program?

Ans: The Working of a Computer

We will now illustrate how the algorithm is executed by a computer. The model of the computer will be used for illustration.
We will assume that the flowchart is first read by the input unit and stored in computer's memory. (Strictly speaking, a program and not a flowchart is read and stored in a computer's memory. We use the flowchart instead of program, only because the reader may not know a programming language.) The data to be processed, namely a set of papers, each with a roll number and marks, are queued up at the input unit in the order in which they would be used by the flowchart.
The processing unit of the computer reads from the memory the first block in the flowchart, which says `START'. This instruction gets all the unit in computer ready. After this, the processing unit retrieves from the memory the next instruction given in the flowchart which states:

READS ROOL NO. MARKS
This commands is interpreted as "Read the first number from the data record waiting at the top of the queue at the input unit. Label a box in memory as ROLLNO. Clear it and store in it the number read. Similarly store the second number from the data record in a memory box labeled MARKS.
The next is now taken up for execution. It is interpreted as: "clear a memory box and label it MAXROLLNO. Copy into it the contents of memory box ROLLNO. Clear another memory box and label it MAXMARKS. Copy into it contents of box MARKS".

Following the flowchart, the next block (block 4) asks the question: "Any more papers?" This is interpreted by the processing unit and it checks if there is any more papers queued up at the input unit. Papers 2 and 3 would still be there and answer to the question is "Yes". The path labeled "Yes". Is thus followed and block 5 in the flowchart is taken up for execution. It states:

READ ROLLNO. MARKS
It is interpreted as before and before and the numbers in the paper waiting at the top of the queue in the input unit, namely paper 2, are read and the values in it are stored in boxes ROLLNO and MARKS after erasing their earlier contents.

b) Describe the hexadecimal representation of numbers?

Ans: HEXADECIMAL

The average number of bits needed to represent a decimal digitis3.2, as pointed out in section 2.2.Thus the binary equivalent of a 10-digit number will be approximately 32 bits long .it is difficult to write such long strings of 1s and s and convert them to equivalent decimal numbers without making mistakes. The hexadecimal system, which uses 16 as base is a convenient notation to express binary numbers. This system by definition uses 16 symbols, viz..,0,1,2,3,4,5,6,7,8,9,A,B,.C,D,E,F. Note that the symbols A,B etc. now represents numbers. As 16 is a power of 2, namely 2 raise power 4, there is a one to one correspondence between a hexadecimal digit and its binary equivalent. We need only 4 bits to represent a hexadecimal digit.

Table 2.5 gives a table of hexadecimal digits and their binary and decimal equivalent.

A binary number can be quickly converted to its hexadecimal equivalent by grouping together successively 4 bits of the binary number starting with the least significant bit and replacing each 4 bit group with its hexadecimal equivalent given in table 2.5.The examples below illustrate this.
Examples

1. Binary number 0111 1100 1101 1110 0011
   Hexadecimal equivalent: 7 C D E 3

2. Binary number 001 0001 1111 0000 . 0010 1100
   Hexadecimal equivalent: 1 1 F 0 . 2 C

   Observe that in example 1.2.above, groups are format from left to right for the fractional part of the number and from right to left for the integer part. If the number of bits in the integer part is not a multiple of 4,we insert leading 0s,as leading 0s have no significance for the integer part. If the number of bits in the fractional part is not a multiple of 4, then we introduce trailing 0s,as trailing 0s have no significance in the frictional part.
   Conversion from hexadecimal to decimal system is simple. It uses the fact that the base of the hexadecimal system is 16.We give below two examples of hexadecimal to decimal conversion.

Q3. (a) Discuss computer input units?

Ans: FLOPPY DISK INPUT UNIT:-

Data recorded on a floppy disk is read and stored in a computer memory by a device called floppy reader. A floppy disk is inserted in a slot of floppy input unit. The disk is rotated at around 360 revolutions per minute.

Magnetic ink character recognition:-
In this method, human readable characters are printed on documents using special magnetic ink. A magnetic ink character reader can recognize such characters. In a cheque, for instance, the branch code, account number and cheque number are preprinted at the bottom using magnetic ink.

Optical mark reading and recognition:-
In this method special preprinted forms are designed with boxes, which can be marked with dark pencil or ink. Each box is annotated distinctly so that the user understands what response he is marking.

Optical character recognition: -
An optical scanne0r is a device is used to read an image, convert it into a set of 0 and 1 and stored this computer memory. The image may be a hand written document, a typed or a printed document or a picture.

Bar coding:-
In this method small bars of varying thickness and spacing are printed on packages, badges, tags etc. which are read by optical readers and converted to electrical pulses. The pattern of bars is unique and standardized in some countries. e.g in USA each grocery product has been given a unique ten digit code and this is represented in bar code form on every container of product.

Speech input unit:-
A unit, which takes as its input spoken words and converts them to a form which can be understood by a computer is called a speech input unit. By understanding we mean that the unit can uniquely code (as a sequence of bits) each spoken word and can interpret and initiate actions based on the word.

b) Discuss the compiling of high-level language program?

Ans: COMPILING HIGH LEVEL LANGUAGE

In this section we will discuss briefly the steps in compiling a high-level language program to an executable machine language program. Broadly the compilation process of two steps. The first step is the analysis of the source program and the second is the synthesis of the object program in the machine language of the specific machine. The analysis step uses the precise description of the source programming language. A source language is description of source programming language. A source language is described using lexical rules, syntax rules and semantic rules.
Lexical rules specify the valid syntactic elements or words of the language. Syntax rules how the valid syntactic elements are combined to form statements of the language. Semantic rules are assign meaning to valid statements of the language.

The steps used in the translating a high level language source program to executable code is given. The first block is a lexical analyzer (or scanner). If reads successive lines of the program and breaks then into lexical items, namely, identifier, operator, delimiter etc., and attaches a type tag to each of these. Besides this it construct a symbol table for each identifier and finds the internal representation of each constant. The symbol table is used latter to allocate memory to each variable.

The second stage of translation is called syntax analysis of parsing. In this phase expressions, statements, declarations etc., are identify by using the results of lexical analysis. Syntax analysis is aided by using the technique based on formal grammar of the programming language.
In the semantic analysis units recognized by the syntax analyzer are processed. An intermediate representation of the final machine language code is produced. This phase bridges the analysis and synthesis phases of translation.
The phase of translation is code generation. A number of optimization to reduce the length of machine language is carried out during this phase. The output of the code generator is the machine language program of specific computer. If the sub program library is used or if some subroutines are separately translated a final linking and loading step in needed to produce the complete machine language program ready for execution. If subroutines are separately compiled the addresses of the resulting machine language instruction will not be there final address when all the routines are placed together in main memory. The linkers' job is find the correct location of the executable program. The loader will then place then in the memory at their right addresses.

c) Write a note on the layer of Unix operating system?

Ans: The UNIX OS can be thought of as a layered system. The innermost core is called the Kernel. Kernel is a set of programs, which perform various primitive operations requested by user process. The following services are provided by the UNIX Kernel.
Controlling the creation, suspension and termination of process. Inter-process communication.
Scheduling process on the CPU. In a time-shared mode the CPU is shared by several process. Thus each process is given a time slice.
Main memory is allocated to an executing process as requested by the process. If sufficient memory is not available a waiting process is written on a disk and memory occupied by it is temporarily allocated to the executing process.
It manages users files by providing them space in secondary storage, protecting them from illegal access and ensuring efficient storage retrieval.
It provides processes controlled access to peripheral devices such a terminals, disk drives, network devices etc.
All these services are provided by hiding all low-level details from users process.
The next layer has programs supported by the OS. The most interesting of these is one called shell, which is a command interpreter.
This file can be made an executable file and when invoked will act like a new shell command. UNIX also provides a very interesting idea called a pipe. A pipe is a way to send the output of one program as the input of another program without restoring. There are large families of UNIX shell programs that read an input, perform single transformations, & write an output. For example there is a command called grep, which searches a file for lines that match a given pattern that outputs them.

The outermost layer of UNIX has language compilers for C, FORTRAN 90 etc. A unique feature of the language compilers in UNIX is that it provides a common object code format, which allows easy mixing of high-level languages.
Executions of user processes on UNIX systems is divided into two modes known as user mode & kernel mode. When a user process requests services provided by the UNIX kernel the execution mode changes from the user mode to the kernel mode.
Further, there are some special hardware instructions, which can be executed only in kernel mode. It should be remembered that the kernel runs on behalf of a user process & does not function independently. Processing interrupts & exceptions & management of memory are all delegated to the kernel.

Q4. (a) Define minimum microcomputer configuration in brief?

Ans: A Minimum Microcomputer Configuration:-

Along with a microprocessor chip, other support chips are required to configure a microcomputer system.
At the very minimum it necessary to have the following chips:

(1) A voltage regulator chip used along with a full wave rectifier connected to the mains.
This provides a power Supply regulated to +_5%.

(2) A clock signal, which is a square wave, is required to synchronize the operations of a microprocessor. A special clock chip is used along with quartz of appropriate type to generate this clock.

(3) Integrated circuit chips to buffer and control the information from memory to the processor and from input/output devices to the processor.

(4) Memory chips and associated controller to interface them to the processor.

The trend is to reduce the need for support Chips and build controllers within the processor.

(b) What is the need of computer communication Network?

Ans: Need For Computer Communication Networks

We will discuss in greater detail the need for each of the types of computer communication. Remote time sharing terminals are most useful for program development. The rapid turnaround provided by such a use increases programmer productivity. A user who has a personal computer at home would use it for most of his work and connect the video terminal of the personal computer to a big computer for solving larger programs and to access special library programs and data resident in the big computer.

Another use is by smaller organization, which may not have the workload to justify an in-house computer. In such a case they buy a workstation or a pc, place it in their premises and connect. It as a time-sharing terminal to a larger computer via a telephone line. This allows them to conveniently access a larger machine without having to make frequent trips to the computer center.
Another important remote terminal application is for information retrieval. Some information centers store large amounts of data on patents, technical reports, journal articals,etc.,in an
Organized fashion. A user requiring specific information, say on patents in a specified area can connect his terminal through a telephone line to a large computer and retrieve the information using appropriate descriptors. Some information centers are connected to the International fax networks and it would be possible to send enquiries via fax to such centres. Rapidly many such centers are being connected to Internet. Enquiries and replies would then be by electronic mail.
As we saw in the last section, localarea networks are used to interconnect many computers within an organization. The purpose of interconnection would be to share files, share programs, and decentralized specialized function. Another reason for creating a local network is also to share the use of expensive peripheral such as fast printers, large disks, graphics workstations,etc. Similar local networks are useful in a laboratory environment where each sophisticated instrument has a built-in microprocessor. These can be interconnected and the network connected in turn to a general purpose computer with powerful I/O devices and storage devices. The general-purpose computer and the peripheral devices enhance the power for analyzing the output of each of the instruments. Besides this, data gathered and processed by each instrument may be correlated.

Local area networks are also used in factories for controlling plants and processes. Individual small computers would be usually installed to monitor and control critical processes in the plant. These computers may be interconnected and connected in turn to another computer, which would perform supervisory functions. Such a network provides an integrated control of the plant.
The communication lines interconnecting the computers in LAN are short. It is also localized to “private” area and one need not use a public telephone network. As distances are small and as faster communication between processors in the LAN is desirable, high-speed communication lines which can transmit around ten million to hundred million bits per second are used to interconnect them.
Computer networks are mainly used to connect a number of widely dispersed computers. The main objective of such an interconnection is to allow users of the network to access specialized library programs, databases, languages and special facilities available in any of the computers in the network. For example, it would not be possible for many organizations to install a supercomputer, which may cost 15 million dollars. If a supercomputer is connected to a network then it is possible for many organizations to access it form their location.

Q5. Discuss
i) Keyword and identifiers.

Ans: KEYWORDS AND IDENTIFIERS

Every C word is classified as either a keyword or an identifier. All keywords have fixed meanings and these meanings cannot be change. Keyword serves as basic building blocks for program statements. The list of ANSI C is listed in table 2.3.All keywords must be written in lowercase .Sime compilers may use additional keywords that must be identified from the C manual.

Table 2.3 ANSI C Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

identifiers refer to the names of variables. functions and arrays. These are defined names and consist of a sequence of letters and digits. with a letter as a first character. Both uppercase and lowercase letter are permitted, although lowercase letters are commonly used. The underscore character is also permitted in identifiers. it is usually used as a link between two words in long identifiers.

ii) Bitwise operators.

Ans BITWISE OPERATOR

C has a distinction of supporting special operators known as bit wise operator for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right to left. Bitwise operators may not be applied to float or double. Table 3.5 lists the bitwise operators and their meaning.

Table 3.5 Bitwise Operators

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right

iii) Formatted input.

Ans: Formatted Input
Formatted input refers to an input data that has been arranged in a particular format. For example, consider the following data:
15.75 123 john
This line contains three pieces of data, arranged in a particular form. such data has to be read conforming to the format of its Appearance. For example, the first part of the data should be read into a variable float, the second into int, and the third part into char. this is possible in C using the scanf function.(scanf means scan formatted.)
We have already used this input function in a number of examples. Here, we shall explore all of the options that are available for reading the formatted data with scanf function. The general form of scanf is
scanf("control string",arg1,arg2,.......argn);

The control string specifies the field format in which the data is to be entered and the arguments arg1, arg2.......argn specify the
Address of location where the data is stored. Control string and arguments are separated by commas.
Control string (also know as format string) contains field specifications, which direct the interpretation of input data. It may include:
1. Field (or format) specifications, consisting of the conversion character%, a data type character (or type specifier), and an optional number, specifying the field width.
2. Blanks, tabs, or newlines.
Blanks, tabs and newlines are ignored. The data type character indicates the type of data that is to be assigned to the variables associated with the corresponding argument. The field width specifier is optional.

iv) Formatted output.

Ans Formatted Output
We have seen the use of printf function for printing captions and numerical results. It is highly desirable that the output are produced in such a way that they are understandable

and are in an easy-to-use from. It is therefore necessary for the programmer to give care-full consideration to the appearance and clarity of the output produced by this program. The printf statement provides certain features that can be effectively exploited to control the alignment and spacing of print-outs on the terminals. The general form of printf statement is
printf("control string", arg1, arg2,......,argn);

Control strings consist of three types of items;
1. Characters that will be printed on the screen as they appear.
2.Format specifications that define the output format for display of each time.
3. Escape sequence character such as \n, \t, and \b.
The control string indicates how many arguments follow and what their types are. The arguments arg1, arg2,........., argn are the variable whose values are formatted and printed according to the specification.
A simple format specification has the following form:
% w.p type-specifier where w is an integer number that specifies the total number of digital of columns for the output values and p is another integer number.

Q6. (a) Describe the else if ladder with example?

Ans: THE ELSE IF LADDER
There is another way of printingifs together when multipat decisions are involved .a multipath decision is a chain of ifs in which the statements associated with each else is an if. It takes the following general form:

```
if ( condition 1)
statement -1;
else if ( condition 2)
statement -2;
else if ( condition 3)
statement -3;
else if ( condition n)
statement -n;
else
default - statement;
statement -x;
```

This construct is known as the else if ladder. The conditions are evaluated from the top (of the ladder), downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement x (skipping the rest of the ladder). When all the n conditions become false, then the final else containing the default statement will be executed.
Let us consider an example of grading the students in an academic institution. The grading is done according to the following rules:

Average marks	Grade
80 to100	Honors
60 to 79	first division
50 to 59	Second division
40 to 49	Third division
0 to 39	Fail

this grading can be done using the else if ladder as follows:

```
if (marks >79)
grade ="honours";
else if (marks > 59)
grade ="First division";
else if (marks >49)
grade ="Second division";
else if (marks >39)
grade ="third division";
else
grade ="Fail";
printf ("%s\n",grade);
```

consider another example given below :

```
------
------
------
if(code ==1)
colour ="RED";
else if (code ==2)
colour ="GREEN";
else if(code =3)
colour ="WHITE";
else
colour ="YELLOW";
------
------
------
```

code numders other than 1,2 or 3 are considered to represent YELLOW colour . the same results can be obtained by using nested if...else statements .

b) Explain *Do-While* statement with examples?

Ans: The Do-While loop construct that we have discussed in the previous section makes a test of condition before the loop is executed. Therefore, the body of loop may not be executed at all if the condition is not satisfied at the very first attempt on some occasions

It might be necessary to execute the body of the loop before the test is performed. Such situation can be handled with the help of do statement. This takes the form;

```
do
{
body of the loop
}
while (test-condition);
```

on reaching the do statement, the program proceeds to evaluate the body of the loop first. at the end of the loop, the test condition in the while statement is evaluted.if the condition is true the program continues to evaluate the body of the loop once again. This process continues as long as the condition is true. When the condition become false, the loop will be terminated and the control goes to the statement that appears immediately after the while statement.

since the test - condition is evaluated at the bottom of the loop, the do...while construct provides an exit- controlled loop and therefore the body of the loop is always executed at least once.

A sample example of a do while loop is:

```
............
do
{
print f("input a number\n");
number = getnum ( );
}
while(number > 0);
```

This segment of a program reads a number from the keyboard until a zero or a negative number is keyed in, and assigned to the
Sentinel variable number.

The test condition may have compound relation as well. for instance, the statement

while (number > 0 && number < 100):

in above example would cause the loop to be executed as long as the number keyed in lies between 0 and 100

Consider another example:

```
---------------
I = 1:
sum = 0;
do
{
sum = sum + I;
I = I+2;
}
while(sum < 40 | | I < 10);
printf("%d%d\n", I, sum);
----------------
```

The loop will be executed as long as one of the two relation is true.

Q7. (a) Discuss the initialization of two 'dimensional arrays'?

Ans:

Like the one dimensional arrays, two-dimensional arrays may be initialized by following their declaration with a list of initial values enclosed in braces. For example,

```
int  table[2][3] ={0,0,0,1,1,1};
```

Initializes the elements of the first row to zero and the second row to one. The initialization is done row by row. The above statements can be equivalently written as

```
int table[2][3] ={0,0,0} ,{1,1,1};
```

By the surroundings the elements of the each rows by braces.

We can also initialize a two dimensional array in the form of a matrix as shown below:

```
int table [2][3] = {
{0, 0, 0,}
{1, 1, 1}
};
```

Note the syntax of the above statements. Commas are required after each brace that closes off row, except in the case of the last row.

When the array is completely initialized with all values, explicitly, we need not specify the size of the first dimension. That is, the statement

```
int table [ ][3] = {
{0,0,0}
{1,1,1}

};
```

Is permitted

If the values are missing in an initializer, they are automatically set to zero. For instance, the statement

```
Int table [2] [3]={
{1,1}
{2}
};
```

Will initialize the first two elements of the first row to one, the first element of the second row to two and all other elements to zero. When all the elements are to be initialized to zero, the following short cut method may be used.

int m [3][5] = {{0}, {0}, {0}};
The first element of each row is explicitly initialized to zero while the other elements are automatically initialized to zero. The following statement will also achieve the same result;
int m [3][5] ={0, 0};

b) How can you write the string to the screen?

Ans: WRITING STRINGS TO SCREEN

Using printf Function

We have used extensively the print function with %s format to print strings to the screen. The format %s can be used to display an array of characters that is terminated by the null character. For example, the statement
printf("%s", name);

Can be used to display the entire contents of array name.
We can also specify the precision with which the array is displayed. For instance, the specification %10.4 indicates that the first four characters are to be printed in a field width of 10 columns.
However, if we include the minus sign in the specification (e.g., %-10.4s), the strings will be left -justified. The example 8.4 illustrates the effect of various %s specifications.

Example
Write a program to store the string "United Kingdom" in the array country and display the string under various format specifications.

The output illustrates the following features of the %s specifications.

1. When the field width is less then the length of string, the entire string is printed.
2. The integer value of the right side of the decimal point specifies the number of characters to be printed.
3. When the numbers of characters to be printed is specified as zero, nothing is printed.
4. The minus sign in the specifications causes the strings to be printed left -justified.
5. The specification %. ns prints the first n characters of the string.

Program
main()
{

```
char country[15]="United Kingdom";
printf("\n\n");
printf("*123456789012345*\n");
printf("------");
printf("%15s\n,country");
printf("%5s\n",country);
printf("%15.6s\n",country);
printf("%-15.6s\n",country);
printf("%15.0\n",country);
printf("%.3s\n",country);
printf("%s\n",country);
printf("-----");
}
```

Output

```
*123456789012345*
------
United  Kingdom
United  Kingdom
United
United
Uni
United Kingdom
-----
```

Q8. Discuss

i) No argument but no return value with example.

Ans: NO ARGUMENTS AND NO RETURN VALUES

When a function has no arguments, it does not receive any data from the calling function. Similarly, when it does not return a value, the calling function does not receive any data from the called function. In effect, there is no data transfer between the calling function and the called function.
Program:

```
/*Function declaration*/
void printline (void);
void value(void);

main()
{
printline();
value();
printline();
}
/*              function1:printline()             */
Void printline(void)       /* contains no argument*/
{
int I ;
for(i=1; i<=35;i++)
printf("%c",'_');
printf("\n");
}
/*      function2:  value()      /*
void value(void)              /*contains no argument*/
{
int   year, period;
float   inrate, sum, principal;

printf("principal amount?");
scanf("%f", &principal);
printf("interest rate? ");
scanf("%f", &inrate);
printf("period?  ");
scanf("%d", &period);
sum=principal;
year=1;
while(year<=period);
{
sum = sum*(1+inrate);
year= year+1;
}
printf("\n%8.2f %5.2f %5d %12.2f\n", principal, inrate, period, sum);
}
```

```
OUTPUT:
Principal  amount?    5000
Interest  rate?          0.12
Period?                     5

5000.00    0.12       5      8811.71
```

As pointed out earlier, a function that does not return any value cannot be used in an expression can only be used as an independent Statement

ii) Function that return multiple values.

Ans: Functions that return just one value using a return statement. That is because; a return statement can return only value. Suppose, however, that want to get more information from a function. We can achieve this in C using the argument not only to receive information but also to send back information to the calling function. The argument that are used to "send out" information are called output parameters.
The mechanism of sending back information through argument is achieved using what are known as the address operator <&> and indirection operator <*>. Let us consider an example to illustrate this.

```
Void mathoperation ( int x, int y, int *s, int *d);
Main ( )
{
Int x = 20, y=10, s,d;
Mathoperation (x,y,&s,&d);

Print f ("s=%d\n d=%d\n'', s, d);
}
Void mathoperation (int a, int b,int *sum, int *diff)
{
*sum=a+b;
*diff =a-b;
}
```

The actual arguments x and y are input argument, s and dare output argument. In the function call, while we pass the actual values of x and y to the function, we pass the addresses of locations where the values of s and d are stored in the memory. (That is why, the operator & is called the address operator).

Q9 a) Discuss declaration of pointer variable with example

Ans: Pointer variables are declared similar to normal variables except for the addition of the unary * operator. This symbol can appear anywhere between the type name and the printer variable name. Programmers use the following styles:

```
int * p;    /*style 1 */
int * p;    /*style 2 */
int * p;     /*style3*/
```

However the style 2 is becoming increasingly popular due to the following reasons:

1. This style is convenient to have multiple declarations in the same statement.
   Example:
   int * p ,x,*q;

2. This style matches with the format used for accessing the target values.
Example:

```
Int  x,*p ,y;
X=10;
P=&x;
Y= *p;     /* accessing x through p*/
*p=20;    /* assigning 20 to x*/
```

We use in this book the style 2,namely,
Int*p;

Q9 b): Describe the following

i) getc and putc functions

Ans: The simplest file i/o functions are getc and putc. These are analogous to getchar and putchar functions and handle one character at a time. Assume that a file is opened with mode w and file pointer fp1. Then, the statement
Putc(c, fp1);
Writes the character contained in the character variable c to the file associated with FILE pointer fp1. Similarly, getc is used to read a character from a file that has been opened in real mode.
For example, the statement
c=getc(fp2);
Would read a character from the file whose file pointer is fp2.
The file pointer moves by one character position for every operation of getc or putc. The getc will return an end-of-file marker EOF, when end of the file has been reached. Therefore, the reading should be terminated when EOF is encountered.

ii) fprintf and fscanf functions

Ans:- So far, we have seen functions, which can handle only one character or integer at a time. Most compilers support two other functions, namely fprintf and fscanf, that can handle a group of mixed data simultaneously.
The functions fprintf and fscanf perform i/o operations that are identical to the familiar printf and scanf functions, except of course that they work on files. The first argument of these functions is a file pointer which specifies the file to be used. The general form of fprintf is
fprintf (fp, ”control string” ,list);

Where fp is a file pointer associated with a file that has been opened for writing. The control string contains output specifications for the items in the list. The list may include variables, constants and strings.

Example:
fprintf(f1,”%s %d %f”, name,age,7.5);
Here, name is an array variable of type char and age is an int variable.
fprintf(fp,”control string”, list);
This statement would cause the reading of the items in the list from the file specified by fp,according to the specifications contained in the control string.

Example:
fscanf(f2,”%s %d”, item, &quantity);
Like scanf , fscanf also returns the number of items that are successfully read. When the end of file is reached, it returns the value EOF.

TEXTBOOK

I. Fundamentals of Computers, V. Rajaraman, Fourth Edition, PHI, 2007

II. Programming in ANSI C, E. Balagurusamy, Third Edition, Tata McGraw Hill