Solved Examination Questions

1. What are computer programs and computer programming? Explain the steps that are required to build a program for solving a certain problem. **[2074 Ashwin]**

Ans: The set of instructions that a computer follows in order to solve the particular task are called as computer programs.

Computer programming is the process of developing and implementing various sets of instructions to enable a computer to do certain task.

Different steps required for solving a certain problem in computer system are:

(i) Problem analysis:

It is the process of understanding and analyzing the program, getting familier with it and dividing it into several modules to write program easily. Here, input and output variables are separated and processing part is linked with them.

(ii) Algorithm development:

It is the step by step method of solving the program. The steps includes the descriptions and sequential method to solve a problem. It is written in simple english language.

(iii) Flowchart designing:

It is the pictorial representation of algorithm using various symbols, connectors and arrows. It helps to represent the program logics and procedures in graphical format so that program can be understood and analyzed easily. Hence it helps in effective coding and program maintenance.

(iv) Coding:

Coding is the process of writing a program in computer understandable format. Here, paper works like algorithm and flowcharts are converted into sequence of codes having standard format. The written code is called source code. It is written using programming language like QBASIC, JAVA, C etc.

(v) Compilation and Execution:

(vi) Debugging and Testing:

It is the process of find the errors (syntax as well as semantic) and removing them. Sometimes wrong output may occur even if the program executes successful due to logical errors. Hence in this process these errors are removed. i.e. debugging is done.

(viii) Program documentation:

It is the collection of information about the program so that it can be understood easily by other programmers in future.

Diagrams such as flowchart, data flow diagrams, comments etc. are various techniques of program documentation.

2. What do you mean by programming language? Explain about the evolution of programming languages. **[2073 Shrawan]**

Ans: The language which specifies a set of instruction to instruct computers to perform the specific task is known as programming language.

The evolution of programming language can be explained in terms of generations of programming language.

(I) First Generation Programming Language:

Also known as machine level language, it is written in the form of binary numbers consisting of 0 and 1. The code written is directly understood by the machine and the language translator is not required. However writing codes in binary language is tedious and difficult.

(II) Second Generation Programming Language:

Also known as assembly language, it uses mnemonics of machine language instead of binary language.

e.g. ADD - addition
INR - Increase
SUB - Subtraction etc.

The program written in assembly language is converted to machine language by assembler. It is machine dependent language.

(III) High level Programming Language:

It uses English language to write the program and hence programming is easier, faster more convenient. It is machine independent, hence is portable language. The codes written are translated to machine language by set of programs called compiler or interpreter. E.g. PASCAL, QBASIC, JAVA etc.

3. What are the advantages of a flowchart design? Write down the guidelines to be followed to draw a flowchart. **[2073 Shrawan]**

- It provides the clear logic, ideas and description of the program.
- The program can be understood and analyzed effectively.
- Errors can be easily found and hence it helps in debugging process.
- Flowcharts help in proper documentation of program.
- Coding is effective as it acts as blueprint of program.
- Specific language code is not necessary and hence, the logic can be used to write codes in different programming language.

The guidelines to be followed to draw a flowchart are:

- Only standard flowcharting symbols should be used.
- Flowchart begins with start and stops with end or stop
- Arrows are used to show the direction of flow of program.
- Flowchart should be clear and easy to follow and there should be no ambiguity in understand.
- There should be only one flowline emerging from a symbol and in case of decision. Proper/possible answer should have a flowline.
- Plane English language should be used.
- The logic used should be valid for all the cases that may occur in program.

4. Why do we need analysis and design before coding a program?

[2072 Kartik]

Ans: Analysis is the process of breaking down of a program into various parts so as to solve the problem easily. It gives clear and concise ideas to write the program.

We need analysis and design before coding a program because of following reasons:

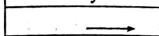
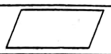
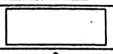
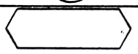
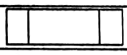
- Programs are well understood and careful procedures are followed. So, there are less chances of getting false outputs and difficulty in coding.
- It gives pre knowledge about what output should come from the system being developed and the methods for developing program.
- Effective program can be generated in shorter and easier possible ways.
- Reduntant or irrelevant information can be removed so that same code need not be repeated in the program.
- It helps to prepare design of the programs layout.

5. What is flowchart? Use various commonly used flow chart symbols. How does a flowchart help computer programming?

[2072 Chaitra, 2070 Chaitra]

Ans: Flowchart is the graphical representation of an algorithm, using standard symbols connectors and arrows. Different informations are written within the flowcharting symbols of different shape. Connectors are used to indicate that the flow of control continues elsewhere (e.g. another page) and arrows are used to show the direction of flow of control.

The various commonly used flowchart symbols are:

Symbol	Purpose
	Directs flow of control
	Represents beginning and end of task
	Used for input and output
	Used for writing operations
	Used in decision making
	Connect different flow lines
	Indicates for loop
	Used for function call

Flowchart helps in computer programming by following ways:

(Refer to 2073 Shrawan on Page no. 9 (advantages of flowchart))

6. What is a program? Briefly describe types of a computer software.

[2072 Chaitra]

Ans: The set of instructions that a computer follows in order to solve a particular task is called a program. It directs the internal operations of a computer system such as controlling input and output devices as well as managing storage spaces within it.

Computer software are generally classified as system software and application software.

(i) System software:

It is a collection of programs designed to operate, control and extend the processing capabilities of the computer itself. It

comprises of the programs written in low-level languages, which interact with the hardware at the very basic level. It acts as the interface between the hardwares of computer and users. E.g. Operating system, compilers, interpreter etc.

(ii) **Application software:**

Application software is needed to meet the specific requirements of the user. It consists of single program as well as collection of programs (software package which work together to perform specific task.) E.g. MS word, MS excess, database, student record software etc.

7. **Categorize programming languages on the basis of their uses and applications. Among them which programming language is C-Programming language.** [2071 Shrawan]

Ans: On the basis of uses and applications programming languages are classified as: procedure oriented language and object oriented language.

(i) **Procedure oriented language:**

These are used to express the logic or the procedure of the program i.e. instruction in step by step sequence. It uses one or more blocks of statement to perform a certain function. So, it is also called as modular programming language.

(ii) **Object oriented language:**

It enables to solve the specific problems or develop certain applications by enabling user to describe what we want. It is non modular programming language.

C-programming is procedure oriented programming language.

8. **List and define different steps to solve the problem in computer system.** [2070 Ashad]

Ans: Different steps to solve problems in computer system are:

(i) **Problem analysis:**

It is the process of understanding and analyzing the program, getting familier with it and dividing it into several modules to write program easily. Here, input and output variables are separated and processing part is linked with them.

(ii) **Algorithm development:**

It is the step by step method of solving the program. The steps includes the descriptions and sequential method to solve a problem. It is written in simple english language.

(iii) **Flowchart designing:**

It is the pictorial representation of algorithm using various symbols, connectors and arrows. It helps to represent the program logics and procedures in graphical format so that program can be understood and analyzed easily. Hence it helps in effective coding and program maintenance.

(iv) **Coding:**

Coding is the process of writing a program in computer understandable format. Here, paper works like algorithm and flowcharts are converted into sequence of codes having standard format. The written code is called source code. It is written using programming language like QBASIC, JAVA, C etc.

(v) **Compilation and Execution:**

Source code is changed into object code after compilation process. Compilation is done using language translators or compilers. This process is successful only if no syntax errors are present.

(vi) **Debugging and Testing:**

It is the process of find the errors (syntax as well as semantic) and removing them. Sometimes wrong output may occur even if the program executes successful due to logical errors. Hence in this process these errors are removed. i.e. debugging is done.

(viii) **Program documentation:**

It is the collection of information about the program so that it can be understood easily by other programmers in future.

Diagrams such as flowchart, data flow diagrams, comments etc. are various techniques of program documentation.

9. **Define software. Explain its types.** [2070 Chaitra]

Ans: Software is defined as the collection of different programs that directs the computer to perform certain tasks.

Types

Refer to 2072 Chaitra on page no. 11

- Comma operators are used to link related expression together.
  e.g. int p, q, r, s, t;
  float m, n, o; etc.
- Sizeof ( ) operator is used to give the size that the operand uses in its memory location.
  e.g. int q = 8;
  int r = size of (q);

- **Precedance of operators:**
  If more than one operators are involved in an expression, C language has a predefined rule of priority for the operators. This rule of priority of operators is called operator precedance.
- **Associativity of operators:**
  If two operators of same precedance (priority) is present in an expression, associativity indicates the order in which they execute.

Solved Examination Questions

1. Explain with an example the role that precedance and associativity play in the execution of an expression. Rewrite the following program by correcting any errors, if present and also write down the output of the corrected code.

```
Define MAX '5'
int main ( )
{
    int case [MAX] = {2, 3, 5, 4, 10}, i, sum=0;
    for (i = 0, i < MAX, i + = 1
    {
    printf ("case %d = %3.2 d\n", i, case [i]);
    sum + = *case + i;
    };

    average = sum /MAX;
    printf ("% 06.2f", average);
    return1;
}
```

[2074 Ashwin]

ns: If more than one operators are used in an expression, the order of evaluation of those operators according to their priority is known as precedance of operator.

E.g. *, /, % operators have higher precedance than +, - operators.

If more than one operators of same precedance appear in an expression, the order in which they execute is known as its associativity.

E.g.: Use of precedance

```
int num1 = 10, num2 = 20;
int result;
result = num1 * 2 + num2;
```

Here, in expression

result = num1 * 2 + num2, multiplication operator has higher precedance than addition and then assignment operator. So, it is evaluated as:

result = 10*2 + 20;
result = 20 + 20;
result = 40;

E.g. Use of associativity

result = num1 * 2 + num2/2;

Here, * and / operators have same priority. So, it is evaluated from left to right (associativity)

result = 10 * 2 + 20/2;
result = 20 + 20/2;
result = 20 + 10;
result = 30;

Rewriting the corrected code,

```
#include <stdio.h>
#define MAX 5

int main ( )
{
float average;
int case [MAX] = {2, 3, 5, 4, 10}, i, sum = 0;
for (i = 0; i < MAX; i + = 1)
{
printf ("case %d = %3.2 d \n", i, case [i]);
sum + = * (cas + i);
```

```
}
average = sum / MAX;
printf ("%06.2f", average);
return 0;
}
```

OUTPUT:
case 0 = 02
case 1 = 03
case 2 = 05
case 3 = 04
case 4 = 10
004.00

2. Explain relational and logical operators. [2073 Shra

Ans: **Relational operators:**

Relational operators in C programming is used for specifying relation between two operands such as greater than, less than equals. We can compare the value stored between two variables depending upon the result, we follow different blocks of statements using relational operators.

Different relational operators used in C are:

Operator	Meaning
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

E.g:
```
int num = 5;
if (num > 0)
printf ("positive");
```
Result: positive

Logical operators:

Whenever we use if statement, we can use relational operator which tells us the result of the comparison. So, if we want to compare more than one conditions, we need to use logical operations.

Thus, logical operators are used to perform logical operation between operands and give 1 for TRUE and 0 for FALSE as output.

Different logical operators used in C are:

Operator	Name
&&	And operator
||	Or operator
!	Not operator

E.g:
```
int num1 = 30, num2 = 40;
int num = 35;
if (num 1 < num && num 2 > num)
printf ("%d lies between %d&%d", num, num1, num2);
```

Result: 35 lies between 30 & 40

3. What do you mean by precedance and associativity of an operator? Explain with suitable example. [2072 Kartik]

Ans: Refer to 2074 Ashwin page no. 27

4. What is an operator, datatype, constant and variable? Define [2072 Chaitra]

Ans: Operator:

An operator is a symbol that tells the compiler to perform special mathematical or logical function.

E.g. Arithmetic operators: +, −, ×, /, %
Logical operators: &&, ||, !
Bitwise operators: |, &, ^
Relational operators: <, >, = = etc.

Data types:

Data types refer to the type of data that a variable can store. It helps to define the size and assists the memory location for the variables.

E.g. integer type (int)
floating point type (float)
character type (char) etc.

Constants:

Constants refer to the fixed values that the program may not alter during its execution. It may be of various types: integer constant, floating point constant, character constant, string constant etc.

Constants are defined as:

 # define identifier value

E.g. # define Pi 3.14

Variables:

Variables are used to refer the names that holds the certain values

It is defined as:

data_type variable-list;

E.g.
```
int  a, b, c;
float p, q, r;
char name; etc.
```

5. What are relational operators and assignment operators? Explain with examples. [2070 Ashad]

Ans: **Relational operators:**

These are used for specifying relation between two operands such as greater than, less than and equal to. Relational operators compare the value stored between two variables and depending upon the result various statements are executed.

E.g. <, >, >=, <=, ==, !=

```
int num -10;
if (num > 0)
printf ("positive");
else
printf ("Negative");
```

Assignment operators:

Assignment operators are used to assign the result of operation to another variable.

Its symbol is '='

Assignment expression is in the form of

identifier = expression;

Where, identifier, generally refers to variable and expression represents constant, variable or complex operation.

E.g. c = a + b;

Here, value of (a + b) is assigned to variable c.

6. Find out final value of a, b and c where following expressions are executed sequentially. [2070 Chaitra]

```
int a = 2, b = 3, c;
a = (b++) + (++ b) + a;
c = a > b ? a : b;
b = (a ++) + (b - -) + a;
c = c ++ * b - - ;
```

Ans: The expressions are executed as following ways

a = 2, b = 3, c;

- In second statement, ++b is prefix operator.
  So, its value becomes b = 4
  ∴ a = 4 + 4 + 2 ⇒ a = 10
- b ++ is postfix operator. So, it increases value of b by 1 i.e. b = 5 in next statement
- In third statement, c = 10 > 5? 10 : 5
  as 10 > 5 is true, c = 10
- In fourth statement, b = 10 + 5 + 10
  b = 25 and a becomes 11 b becomes 24
- In final statement, c = 10 * 24
  c = 240
  Finally, due to postfix operators,
  c = 241 and b = 23
  ∴ a = 11
  b = 23
  c = 241

7. Give output of following program and justify your answer with reason. [2069 Chaitra]

```
# include <stdio.h>
int main ( )
{
int x = 3, y = 5, z = 7;
int a, b;
a = x * 2 + y/5 - z * y;
b = ++ x * (y - 3) / 2 - z ++ * y;
printf ("a = %d", a);
printf ("b = %d", b);
return 0;
}
```

(2) Unformatted Input/Output functions:

The unformatted I/O statements do not specify how the inputs a outputs are carried out. They donot contain any information abo format specifier, field width and any escape sequence.

Character Input and Output functions:

(i) getc ( ) and putc ( ) :

getc ( ) is used to ask for single character and putc ( ) displays sing character.

Syntax:

character_ variable = getc ( );

putc (character_variable);

E.g.

```
char ch;
ch = getc ( );
putc (ch);
```

(ii) getchar ( ) and putchar ( ):

getchar ( ) reads a character from input device.

Syntax:

character_variable = getchar ( );

It makes wait until a key is pressed and then assigns this into characte variable.

putchar ( ) function displays a character to the standard output device

Syntax: *putchar (character_variable);*

(iii) getch ( ), getche ( ) and putch ( )

getch ( ) and getche ( ) are used to read for single character fro keyboard without waiting for enter key to be hit.

Syntax:

character_variable = getch ( );

character_variable = getche ( );

putch ( ) is used to print single character from the character variable.

Syntax:

putch (character_variable);

String input and output functions

gets ( ) and puts ( ):

gets ( ) is used to read all the input characters unless an enter key i pressed. While space characters special characters, etc. all can be read using gets ( )

Syntax:

gets (string_variable);

puts ( ) is used to print the string value, stored in the string variable.

Syntax: *puts (string_variable)*

Solved Examination Questions

1. Write a C program to generate following pattern using unformatted input/output functions only.

```
            N
          e e e
        P P P P P
      a a a a a a a
    L L L L L L L L L
```

[2074 Ashwin]

Ans:

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

int main ( )
{
char str [ ] = "NepaL";
int i, j;

for (i = 0; i < 5; i++)
{
for (j = 5; j > 0; j --)
putch (' ');
for (j = 0; j < (2 * i + 1); j++)
putch (str [i]);
putch ('\n');
}
return 0;
}
```

2. Write a C program to display following pattern without using formatted input/output statements. **[2073 Shrawan]**

```
Programming
rogrammi
ogrammi
gramm
ram
a
```

Ans:

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

int main ( )
{
char str [ ] = "programming";
int i, j, length;
length = strlen (str);
for (i = 0, i < 6; i ++)
{
for (j = i; j < length - i; j ++)
{
putchar (str [j]);
}
putchar ('\n');
}
getch ( );
return 0;
}
```

3. Define and write syntax of following: **[2072 Chaitra]**

Ans:

(i) gets ( ):

It is used to read all input characters including white spaces as well unless an enter key is encountered. It is unformatted input function.

Syntax: *gets (string_variable)*

(ii) putchar ( ):

It is an unformatted output function used to display single character.

Syntax: *putchar (character_variable)*

(iii) scanf ( ):

It is a formatted input function that is used to read the input characters unless white space is encountered.

Syntax: *scanf ("format", arg1, arg2, ...);*

(iv) strlen ( ):

it is used to count the number of characters of the string.

Syntax:

integer_variable = strlen (string_variable);

4. Write syntax, example and use of following:

(i) printf ( )
(ii) scanf ( )
(iii) getche ( )
(iv) getch ( ) **[2071 Shrawan]**

Ans:

(i) printf ( )

Syntax:

printf ("format", arg1, arg2, ...)

E.g.

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

It is used to display the values on the screen.

(ii) scanf ( )

Syntax:

scanf ("format", arg1, arg2, ...);

E.g.

```
char name [15];
printf ("enter your name:");
scanf ("%s", name);
printf ("your name %s", name);
```

- It is the formatted input function used to take input from uses.

(iii) getche ( )

Syntax:

char_variable = getche ( );

E.g.

```
char sex ;
printf ("enter your sex: M/F");
sex = getche ( );
```

- getche ( ) is used to read single character from keyboard without waiting enter key to be hit. It immediately displays the key pressed.

(iv) getch ( ):

Syntax:

char_variable = getch ( );

E.g.

```
char ch;
printf ("Enter any character:");
ch = getch ( );
printf ("you entered: %c", ch);
```

- It is used to take a character input from user but doesn't display it on screen.

5. **Rewrite the following program by correcting any syntactical error if present. Also show the output of the corrected code.**

```
# include <stdio.h>
int main ( )
{
float root, int i - 1;
do { sum = 2i – 1;
print ("\t %d\ n", sum);
i * = 5/3;
} "while (sum <= 1.5)
root = pow (i, 1/2);
print ("\n% . 3f", root);
return void;
}
```

[2070 Ashad

Ans: The corrected program is:

```
# include <stdio.h>
# include <math.h>
int main ( )
{
float root;
float i = 1, sum = 0;
do { sum = 2*i – 1;
printf ("t %f \n", sum);
i * = 5/3;
} while (sum <=1.5);
root = pow (i, 1/2);
printf ("\n % 3f", root);
return 0;
}
```

6. **What are the differences between formatted and unformatted I/O statements? Describe with proper example.** [2070 Chaitra

Ans: The formatted I/O statements determine the formats in which input and output are executed. They contain the information about format specifiers, field width and any escape sequence. It is more complex form of I/O which requires more space. Here, different inputs taken i.e. characters, integers, floating point numbers are identified separately.

Two formatted I/O functions are:

Example:

```
float num = 2.56784;
char name [15] = "Tribhuwan";
char book [25];
printf ("Enter you favourite book's name: ");
scanf ("% 5\n", &book);
printf ("% . 3f\n", num);
printf ("%10 . 3s\n", name);
printf ("%5\n", Book);
```

Output:

Enter your favourite book's name: Seto Dharti 2.567
Tri
Seto

Thus, output can be modified according to format provided.

Whereas,

unformatted I/O statements do not specify how the inputs and outputs are carried out. They do not contain any information about field width, format specifiers and escape sequence. Different inputs like characters numbers etc. are not distinguished rather they are stored in the form of characters or string only.

Some unformatted I/O functions are:

gets ( ), puts ( ), getch ( ), putch ( ), getche ( ), getchar ( ), putchar ( ) etc.

E.g.

```
Char name [ ] : "Tribhuwan";
Char book [25];
printf ("Enter your fav. book's name : ");
gets (book);
puts (name);
puts (book);
```

Output:

Enter your fav. book's name : Seto Dharti Tribhuwan Seto Dharti

Write a C program to display the following pattern using unformatted output statements:

P
PU
PUL
PULC
PULCH
PULCHO
PULCHOW
PULCHOWK

[2070 Chaitra]

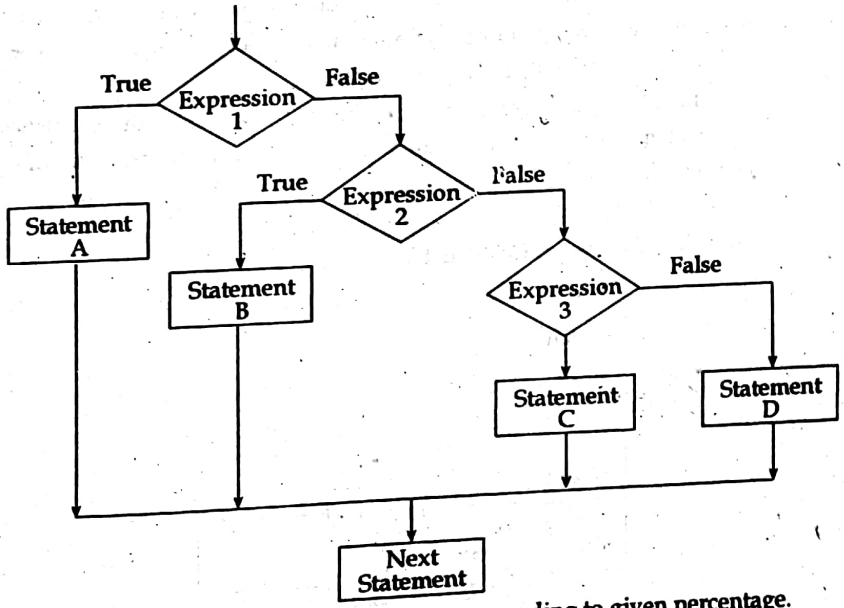
Solved Examination Questions

1. Compare if-else-if ladder and switch construct with example and flowchart. [2074 Ashwin]

Ans: **if.....else .... if ladder**

if-else-if ladder is used for multiway decision in which there is an if-else statement in every else part except the last else. Here, each expression is evaluated one by one in order and when an expression is true, the statements corresponding to the expression is executed and control comes out of nested structure without checking remaining conditions.

Flowchart for if-else-if ladder



A program to find grade of student according to given percentage.

```
# include <stdio.h>
int main ( )
{
float per;
char grade;
printf ("Enter the percentage;");
scanf ("%f", & per);
```

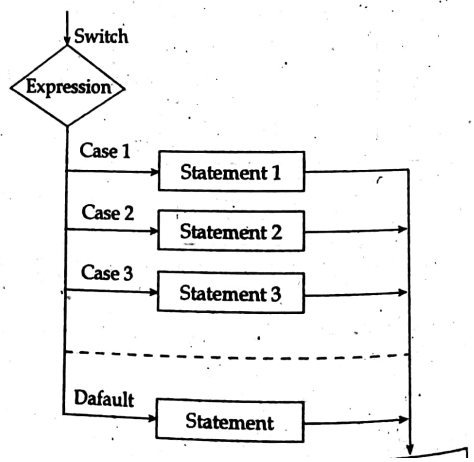
```
if (per >= 80)
grade = 'A';
else if (per >=60)
    grade = 'A';
else if (per >= 40)
    grade = 'C'
else
    grade = 'D'
printf ("Grade is: %c", grade);
return 0;
}
```

Switch Construct:

Switch is a multi-way conditional control statement which helps us make choice among a number of alternatives. Here, an expression tests for equality against the list of variables.

In switch statement, if we want the control to come out of switch case statements after a case evaluated is true, break statement should be used.

Flowchart for Switch Construct:



Example:

Switch case: (a program to do mathematical operation according to given operator)

```
# include <stdio.h>
int main ( )
{
char op;
int a, b;
printf ("Enter number, operator and another
       number.")
scanf ("%d %c %d", &a, &op, &b);
switch (op)
{
case '+':
    printf ("Result = %d/n", a+b);
    break;
case '-':
    printf ("Result = %d /n", a - b);
    break;
case '*':
    printf ("Result = %d \n", a * b);
    break;
case '/':
    printf ("Result = %d \n", a/b);
    break;
    }
return 0;
    }
```

When can we use recursive functions? Why do we need control statements in computer programs? Differentiate between do.....while and for statements.

ns: We can use recursive functions when the problem is to be solved repeatedly breaking it into small problems, which are similar in nature to the original problem. However, there should be a terminating condition for recursion to exit.

Sometimes, we need to use a condition for executing only a part of program or execute some statements several times. Thus, we need control statements to specify the order in which various instructions in

The differences between do-while and for statements are:

For statement	do-while statement
(i) First, the expression is evaluated then, loop body is executed.	(i) First, the loop body executed then the condition expression is evaluated.
(ii) It is entry controlled loop.	(ii) It is exit controlled loop.
(iii) If initial condition is false, the loop body will not execute	(iii) If initial condition is fa loop body will execute onc
Syntax: *for (expression1; expression2; expression3)* *{* *statements* .... *}* *where,* *expression2 is test expression*	Syntax: *do* *{* *statements* .... *} while (test_expression);*

3. Why do we need control statements? Compare switch and if else ladder with example. [2073 Shrawa

Ans: Sometimes, we need to use a condition for executing only a part program or execute some statements several times. Thus, we ne control statements to specify the order in which various instruction the program are to be executed.

Refer to 2074 Aswin on Page no. 55

4. Write a C program to display all characters between a given range. [2073 Shrawa

Ans:
```
# include <stdio.h>
int main ( )
{
int a1, a2, i, swap;
char c1, c2;
printf ("Enter the first and last character:");
scanf ("%c %c", &c1, &c2);
    a1 = c1;
    a2 = c2;
if (a1 > a2)
{
swap = a1;
a1 = a2;
a2 = Swap;
    }
printf ("characters between the given ranges:")
for (i = a1 + 1; i < a2; i++)
    {
print f ("\n %c", i);
    }
return 0;
    }
```


What is the purpose of the continue statement? Within which control statements can continue statement be included? Compare with the break statement. [2072 Kartik]

s: The purpose of continue statement is to go to next iteration of the loop by skipping some statements of the loop at the given condition.
We can include continue statement in looping statements like for, while, do-while. etc.
When continue statement is encountered, the loop is not terminated but skips some statements and starts iteration of other loops.
Whereas,
When break statement is encountered, the loop terminates and control is transferred to another statement following the loop.

Write a program to find the sum of series:
$Sn = \Sigma 1/n^2$ upto N term. [2072 Kartik]

s:
```
# include <stdio.h>
# include <conio.h>
int main ( )
{
int i, n, N;
float Sn = 0;
n = 1;
printf ("Enter the value of N:");
scanf ("%d", & N);

for (i = 1; i < = N; i ++)
```

```
{
Sn = 1/(n*n);
n ++;
}
printf ("\n sum of series: %.3f ", Sn);
return 0;
}
```

7. **How is the switch statement used in decision making? Explain with suitable example.** [2072 Chai

Ans: Switch statement is a multiway conditional statement which helps make choice among a number of alternatives.

Its working mechanism is explained as:

```
switch (expression)
{
case constant 1:
statement
.....
Case constant 2:
statement
.....
.....
case constant N:
statement
.....
default:
statement
.....
}
```

Firstly, switch control expression is evaluated, if the value expression matches with an *case* constant expression then control transferred to the label. If none of *case* constant matches with the value of expression, then the control is transferred to the *default* label; which is optional.

Example:

```
# include <stdio.h>
# include <conio.h>
int main ( )
{
char gen;
```



```
.ntf("Enter your gender: \n 'M' for male, 'F' for
ale, 'N' for none");
inf ("%d", & gen);
switch (gen)
{
case 'M':
printf("Male");
break;
case 'F'
printf("Female");
break
case 'N'
printf ("Neither male, nor female");
break;

default:
printf ("You picked wrong choice");
}
```

8. **Write a program to check whether a given integer is a triangular number or not. (*Any number is a triangular number if the sum of natural number from 1 to any number i is exactly equal to number. For e.g. 1, 3, 6, 10, 15 etc. are triangular numbers as, 1 + 2 + 3 + 4 + 5 = 15*)** [2072 Chaitra]

Ans:

```
# include <stdio.h>
# include <conio.h>
int main ( )
{
int i, num, sum = 0;
printf ("Enter the number to be checked");
scanf ("%d", &num);
i = 1;
while (sum <=num)
{
sum + = i;
i ++;
if (sum = = num)
printf ("The number is triangular");
}
return 0;
}
```

9. **Write the differences between while and do..... while loop and the program to find whether a year is leap or not.** [2071 S...]

Ans: The differences between while and do .... while loop are:

While loop	Do .... while loop
i) First, the expression is evaluated then loop body is executed.	(i) First, the loop body is ex... and then the expres... evaluated.
(ii) If initially condition is false, the loop body will not execute at all.	(ii) Even if the condition a... initially, the loop body execute once.
(iii) It is entry controlled loop *syntax:* *while (test_expression)* *{* *body of loop* ..... *}*	(iii) It is exit controlled loop *syntax:* *do* *{* *Body of loop* *}* ..... *} while (test_expression);*

Program to find whether a year is leap year or not.

```
# include <stdio.h>
# include <conio.h>
int main ( )
{
int year;
print ("Enter a year:");
scanf ("%d", & year);
if (year %4 == 0)
{
if (year %100 == 0)
   {
   if (year %400 == 0)
   printf ("%d is a leap year", year);
   else
   printf ("%d is not a leap year.", year);
   }
   else
   printf ("%d is a leap year", year);
   }
   else
```



```
   printf ("%d is not a leap year", year);
   return 0;
   }
```

10. **Write a program to read the number until 1 is encountered. Also count the number of even number and odd numbers entered by the user.** [2070 Ashad]

Ans:

```
# include <stdio.h>
# include <conio.h>
int main ( )
{
int num, odd, even;
odd = 0;
even = 0;
while (num! = 1)
{
printf ("\n Enter a number");
scanf ("%d", & num);
if (num % 2 == 0)
even ++;
else if (num %d == 1)
odd ++;
}
printf ("No. of odd numbers: %d\n No. of even
    numbers: %d", odd, even);
return0;
}
```

11. **Distinguish between break and continue statement with example.** [2070 Ashad]

Ans: Refer to 2072 Kartik on Page No. 59

Example:

Program using break:

```
# include <stdio.h>
# include <conio.h>
int main ( )
{
int i;
for (i = 1; i < = 10; i ++)
{
```

Solved Examination Questions

1. **Write a program C to find out whether the n^{th} term of the Fibo... series is a prime number or not. Read the value of n from the ... and display the result in the main function. Use separate ... defined functions to generate the n^{th} fibonacci term and to ch... whether the number is prime number is prime or not. [2074 Ashw...**

Ans:

```
include <stdio.h>
int fibo (int);
char prime (int);

int main ( )
{
int n, fib;
char chk;
printf ("Enter value of n:");
scanf ("%d", &n);
fib = fibo (n);
printf ("%dth fibonacci term is : %d", n, fib);
chk = prime (fib);
if (chk = = 'Y')
printf ("%d is prime" , fib);
else
printf ("%d is not prime." fib);
return 0;
}

int fibo (int num)
{
if (num = = 0)
return 0;
else if (num = = 1)
return 1;
else
return (fibo (num-1) + fibo (num-2));
}

char prime (int num)
{
```



```
int i, ctr = 0;

for (i = 2; i < num; i++)
{
if (num %i = = 0)
ctr ++;
}
if (ctr = = 0)
return 'Y';
else
return 'N';
}
```

2. **What are the advantages of using functions? Differentiate between library functions and user-defined functions with suitable example. [2073 Shrawan]**

Ans: Refer to Theory Portion on Page No. 71

The differences between library functions and user-defined functions are:

Library functions	User defined
(i) These are predefined functions in C library.	(i) These are defined by user at the time of writing program.
(ii) We need to use appropriate header files to use these functions.	(ii) Header files are not required.
(iii) They have certain fixed syntax and application. E.g. scanf ( ), printf ( ) strcmp ( ) etc.	(iii) Application and contents can be easily modified. E.g. factorial ( ), sum ( ), area ( ) etc.

3. **Write a program to check whether a given number is armstrong or not using recursive function. [2073 Shrawan]**

Ans:

```
# include <stdio.h>
int armstrong (int);
int main ( )
{
int num, check;
printf ("Enter a number:");
scanf ("%d", & num);
```

```
check = armstrong (num);
if (check = = num)
printf ("Number is armstrong.\n");
else
printf ("Number is not armstrong.");
return 0;
}
armstrong (int n)
{
static int sum = 0;
if (n! = 0)
{
sum + = n * n * n;
armstrong (n% 10)
}
else
return sum;
}
```

4. What is the meaning of function proto typing? Write a program to calculate the sum of series:

1 + 11 + 111 +..... + upto N terms using recursive function.

If N is read as 5, the series is: 1 + 11 + 111 + 1111 + 11111 **[2072 Kartik]**

Ans: Function protyping is a function declaration that specifies the function name and data types of its arguments in the parameter list.

Syntax:

return_type function_name (parameter_list);

E.g. int large (int, int, int);

Program to calculate sum of series using recursion:

```
# include <stdio.h>
int sum (int);
int main ( )
{
int value N;
printf ("Enter the value of N:");
scanf ("%d", N);
value = sum (N);
printf ("sum of series = %d", value);
return 0;
}
int sum (num)
```



```
{
static int sums = 0;
    int add = 0;
    if (num > 0)
    {
    sums = sums * 10 + 1;
    add + = sums;}
sum (num - 1);
else
return add;
}
```

5. Briefly explain the passing by value and passing by function with example. **[2072 Chaitra]**

Ans: Refer to theory on Page no. 78 (Different types of function calls)

6. Write a program to calculate sum of digits of a given 5-digits number entered by the user using recursive function. **[2072 Chaitra]**

Ans:

```
# include <stdio.h>
int sum (int);
int main ( )
{
int num;
printf ("Enter the 5-digit number:");
scanf ("%d", & num);
printf ("\n sum of digits of number: %d", sum(num));
return 0;
}

int sum (n)
{
if (n == 0)
return 0;
else
return (n%10 + sum (n/10);
}
```

7. What is recursive function? How does it work? Find out sum of digit of number until the number becomes one digit number.

[891 > 18 > 9] **[2071 Shrawan]**

Ans: Refer to theory on Page no. 80

```
# include <stdio.h>
int sum (int);
```

```
int main ( )
{
int num;
int value;
printf ("Enter the number:");
scanf ("%d", &num);
value = sum (num);
    while (value > 9)
    {
    value = sum (value);
    }
printf ("sum = %d", value);
return 0;
    }

int sum (n)
{
if (n = = 0)
return 0;
else
return (n% 10 + sum (n/10));
}
```

8. **Explain how function is defined in C? Differentiate call by value and call by reference.** [2070 Ashad]

Ans: Refer to theory portion on Page no. 72 & 78

9. **Write a program using a function that returns the largest number from an array of numbers that is passed to the function.** [2070 Ashad]

Ans:
```
# include <stdio.h>
# define max 100
void large (int [ ]);
int n;
int main ( )
{
int num [max], largest;
int i;
printf ("Enter no. of elements:");
scanf ("%d", &n);
printf ("Enter the elements: \n");
for (i = 0; i < n; i++)
```



```
{
scanf ("%d", num [i]);
}
largest = large (num);
printf ("largest no. =%", largest);
return 0;
}

large (int num [ ])
{
int i; larger = num [0];
for (i =1; i < n; i++)
{
if (num [i] > larger)
larger = num [i];
}
return larger;
}
```

10. **Define "function definition" and write the program to find the sum of two numbers using user-defined functions.** [2070 Chaitra]

Ans: Refer to theory portion on Page no. 72

```
# include <stdio.h>
int sum (int, int);
int main ( )
{
int n1, n2, total;
printf ("Enter two numbers:");
scanf ("%d %d", &n1, &n2);
total = sum (n1, n2);
printf ("sum of two no. = %d", total);
return 0;
}

int sum (int x, int y)
{
return (x + y);
}
```

Solved Examination Questions

1. **How two dimensional arrays are created in C programming? Write a program to read square matrix of size N × N and find sum of both diagonals.** [2074 Ashwin]

Ans: Refer to theory portion on page no. 98

```
#include <stdio.h>

int main ( )
{
int i, j, N, sum1 = 0, sum2 = 0;
int mat [100][100];

printf ("Enter the order of matrix: N");
scanf ("%d", &N);
printf ("Enter the elements: \n");
for (i = 0; i < N; i++)
{
for (j = 0; j < N; j++)
scanf ("%d", & mat [i][j]);
}

for (i = 0; i < N; i++)
{
for (j = 0; j < N; j++)
{
if (i == j)
sum1 + = mat [i][j];

if (i + j = N + 1)
sum2 + = mat [i][j];
}
}

printf ("\n sum of elements of leading diagonal = %d", sum1);
printf ("\n sum of elements of other diagonal = %d", sum2);

return 0;
}
```


2. **Write a program in C to check whether a given string is palindrome or not using user defined function.** [2074 Ashwin]

Ans:

```
#include <stdio.h>

int check (char [ ]);

int main ( )
{
char str [100];
int chk;
printf ("Enter the string to be checked");
scanf ("%s", str);
if (chk = = 1)
printf ("%s is pallindrome", str);
else
printf ("%s is not pallindrome");
return 0;
}

int check (char str1 [ ])
{
char str2 [100];
int i, j, count = 0;
while (str1 [i]! = '\0')
count ++;

for (i = 0; i < count/2; i++)
{
if (str [i]! = str [count -i-1])
{
j = 0;
break;
}
else
j = 1;
return j;
}
```

3. Write a C program to read two strings in main and compare them in user defined function. Display appropriate message from main. [2073 Shrawan]

Ans:

```
# include <stdio.h>
# include <string.h>
int compare (char [ ], char [ ]);
int main ( )
{
char str1 [30], str2 [30];
int chk;
print ("Enter two strings:");
scanf ("%s %s", str1, str2);
chk = compare (str1, str2);
    if (chk == 0)
    printf("%s and %s are same", str1, str2);
    else if (chk == 1)
    printf ("%s comes before %s", str1, str2);
    else
    printf ("%s comes before %s", str2, str1);
return 0;
}

int compare (char name1 [ ], char name 2 [ ]);
{
int i, j, l1, l2;
int chk;
l1 = strlen (name1);
l2 = strlen (name2);
for (i = 0; i < l1 || i < l2; i++)
{
if (name1 [i] == name2 [i])
{
chk = 0;
continue;
}
else if (name1 [i] > name2 [i])
{
chk = 1;
break;
}
else
chk = -1;
}
return chk;
}
```


4. What are over flow and under flow errors in context of array? Write a program to add corresponding elements of two arrays. The result should form a new array. [2073 Shrawan]

Ans:

Overflow error:

If the number of values assigned to the array members are more in number than the array elements, it causes overwrite of data, which is commonly known as over flow error. E.g. char name [5] = "Sundar"; Here, size of array is 5 but there are 6 elements. Thus, one of the element is over written with other.

Under flow error:

If the size of the array is greater then the no. of elements, it doesn't really cause error but occupies some free space in memory for no reason. It is commonly known as under flow error. E.g. int num [4] = {1,2};

Here, size of array is 4 but it contains only 2 elements. Thus, the array occupies extra memory for no reason.

Considering arrays, to be one dimensional,

```
# include <stdio.h>
int main ( )
{
int i, j, arr1 [20], arr2 [20], arr [20], n;
printf ("enter the number of elements:");
scanf ("% d", &n);
printf ("Enter the elements of first array:\n):
for (i = 0; i < n; i++)
scanf ("%d", & arr1 [i]);
printf ("\n Enter the elements of second array:
    \n");
for (i = 0; i < n; i ++)
scanf ("%d", & arr2 [i]);
printf ("New array:");
for (i = 0; i < n; i ++)
printf ("%d", arr [i]);
return 0;
}
```

5. **Explain with an example for compile time initialization of 2D array. Describe how compiler manages according to the number of initializers and size of an array given by user in case of 1D array.** [2072 Kartik]

Ans: The process of assigning certain set of values to the array elements before executing the program. (i.e. compilation time) is known as initialization of array.

The syntax of initialization of 2D array is:

data_type array_name [row size] [col_size] = {val1, val2, val3 .....valN}

E.g. `int mat [3] [2] = {4, 8, 6, 9, 12, 18}`

Or, `int mat [3] [2] = {{4, 8}, {6, 9}, {12, 18}}`

While initializing 1-D array, it is optional to specify the size of array. If the size is omitted during initialization, then the compiler assumes the size of array equal to the number of initializers.

For e.g. `int marks [ ] = {43, 28, 92, 67};`

`float salary [ ] = {24.8, 29.3, 47.6};`

Here, size of marks is assumed to be 4 and salary is assumed to be 3.

During initilization of 1-D array if the number of initializers is less than the size of array, then all the remaining elements of array are assigned to be zero.

for e.g.

`int marks [5] = {99, 79};`

Here,

size of array is 5, while there are only 2 initializers. After initialization the value of the elements are:

marks [0] = 99, marks [1] = 79, marks [2] = 0,

marks [3] = 0, marks [4] = 0

If the number of initializers is more than the size given to compiler, it will show an error.

for e.g. int arr [3] = {4, 6, 8, 9}.

6. **Write a program to read a word from a main function, pass it into a function that will convert all of its characters into upper case if the first character is in lower case and into lower case if the first character is in upper case. Display the converted string from the main function.** [2072 Kartik]

Ans:
```
# include <stdio.h>
# include <string.h>
void conversion (char [ ]);
int main ( )
{
char str [30];
printf ("enter the word:");
scanf ("%s", str);
conversion (str);
printf ("Converted string: %s", str);
return 0;
}

void conversion (char str [ ])
{
int length;
int i;
length = strlen (str);
for (i = 0; i < length; i++)
{
if (str [0] > 'a' && str [0] < 'z')
{
if (str [i] > 'a' && str [i] < 'z')
str [i] = str [i] - 32;
}
else
{
if (str [i] > 'A' && str [i] < 'Z')
str [i] = str [i] + 32;
}
}
}
```

7. **What is a string?** [2072 Chaitra]

Ans: Refer to theory part on page no. 100

8. **Write a program to read a string and rewrite its characters in alphabetical order.** [2071 Shrawan]

Ans:
```
# include <stdio.h>
# include <conio.h>
# include <string.h>
int main ( )
{
int length, i, Y;
```

```
char str [40], temp;
printf ("Enter a string:");
scanf ("%s", str);
length = strlen (str);
for (i = 0; i < length; i++)
{
for (j = 1 + i; j < length; j++)
{
if (str [i] > str [j])
{
temp = str [i];
str [i] = str [j];
str [j] = temp;
}
}
}
printf ("The string arranged alphabetically: \n");
printf ("%s", str);
return 0;
}
```

9. **A multinational company has hired 3 sales persons for marketing/selling 3 different products in Kathmandu. Each sales person sells each of these products. Write a program to read number of each product sold by all sales-persons. Calculate total sells of each item and the total sells of each sales-person. Use arrays.** [2071 Shrawan]

Ans:

```
# include <stdio.h>
int main ( )
{
int i, j,
int mat [3] [3];
int total = sale1 = sale2 = sale3 = 0;
printf ("Enter the number of product sold by:");
for (i = 0; i < 3; i++)
{
printf ("person %d", i+1);
for (j = 0; j < 3; j ++)
scanf ("%d", & mat [i] [j]);
}
for (i = 0; i < 3; i++)
sale1 + = mat [0] [i];
for (i = 0; i < 3; i ++)
```



```
sale2 + = mat [1] [i];
for (i = 0; i < 3; i++)
sale3 + = mat [2] [i];
total = sale1 + sale2 + sale3;
printf ("No. of sale by 1st person = % d", sale1);
printf ("\n No. of sale by 2nd person = % d", sale2);
printf ("\n No. of sale by 3st person = % d", sale3);
printf ("\n Total no. of sales = % d", total);
return 0;
}
```

10. **How are one dimensonal and two dimensional arrays created in C? Explain with examples.** [2070 Ashad]

Ans: Refer to theory portion on page no. 95 & 98

11. **Write a C program to read two matrices from the user, add them and display the result in matrix form.** [2070 Ashad]

Ans:

```
# include <stdio.h>
int main ( )
{
int i, j, m, n;
int mat1 [10] [10], mat2 [10] [10], sum [10] [10];
printf ("Enter the order of matrices:");
scanf ("%d %d", & m, &n);
print ("Enter elements of 1st matrix: \n");
for (i = 0; i < m; i++)
{
    for (j = 0; j < n; j++)
    scanf ("%d", & mat1 [i] [j];
}
printf ("\n Enter the elements of 2nd matrix:\n");
for (i = 0; i < m; i++)
{
for (j = 0; j < n; j++)
scanf ("%d", & mat2 [i] [j]);
}
printf ("sum : \n");
for (i = 0; i < m; i++)
{
for (j = 0; j < n; j++)
{
sum [i] [j] = mat1 [i] [j]. + mat2 [i] [j];
printf ("%d", sum [i] [j]);
```

The function *realloc ( )* is used to change the size of the memory b... if we want to increase or decrease the memory allocated by mall... or *calloc ( )*.

Note:
If ptr is a null pointer, realloc ( ) behaves like malloc ( ) function.

4. **free ( ):**
Syntax: *free (ptr);*
The function *free ( )* is used to release the memory space alloca... dynamically using *malloc ( ), calloc ( )* or *realloc ( )*.

Solved Examination Questions

1. **What are the advantages of using pointer in C programming? Write... program in C to find second largest element from an arra... containing N elements using concept of pointer?** [2074 Ashwin]

Ans: Refer to Theory portion (use of poiner) on page no. 121

```
#include <stdio.h>

int main ( )
{
int sec, N, i, j, swap;
int mat [100];
printf ("Enter the no. of elements < 100");
scanf ("%d", &N);
printf ("Enter the elements : \n");
for (i = 0; i < N; i++)
scanf ("%d", mat + i);
for (i = 0; i < N; i ++)
{
for (j = i + 1; j < N; j ++)
{
if (* (mat + i) < * (mat + j))
swap = * (mat + i);
* (mat + i) = * (mat + j);
* (mat + j) = swap;
}
}
```



```
}
sec = * (mat + 1);
printf ("second largest element of array is : %d",
sec);
return 0;
}
```

2. **If ptr is a pointer user defined type or basic type, how many bytes is ptr incremented when the statement ptr ++ is executed? [2072 Kartik]**

Ans: When the statement, ptr ++ is executed, the number of bytes incremented depend upon the size of the data type for e.g. it increases by 2 bytes for int, 4 bytes for float, 1 byte for char type of data. The value is given by size of ( ) operator.
syntax: *size of (data_type)*.

3. **Write a C program that calls reverse array ( ) to reverse the array and return the array and display the element of reversed array using pointer.** [2072 Kartik]

Ans:
```
#include <stdio.h>
void reversearray (int [ ], int);
int main ( )
{
int n, i, arr [100];
printf ("Enter the no. of elements of array:");
scanf ("% d", &n);
printf ("Enter the elements of array:");
for (i = 0; i < n; i++)
scanf ("%d", &arr [i]);
reversearray (arr, n)
printf ("\n Array in reverse order: \n");
for (i = 0; i < n; i++)
printf ("%d", arr [i]);
return 0;
}

void reversearray (int arr [ ], int n)
{
int i, swap;
for (i = 0; i < n/2; i++)
```

```
{
swap = arr [i];
arr [i] = arr [n - i - 1];
arr [n - i - 1] = swap;
}
}
```

4. Write a program to read a 3 * 3 square matrix, find minimum [illegible] value of a matrix, replace the diagonal elements by the minim[illegible] element and display it using pointer. [2072 Cha[illegible]

Ans:

```
#include <stdio.h>
int main ( )
{
int i, j, minimum;
int mat [3] [3];
printf ("Enter the elements of 3 x 3 matrix: \n");
for (i = 0; i < 3; i ++)
{
for (j = 0; j < 3; j++)
scanf ("%d", *(mat + i) + j);
}
minimum = ** mat;

for (i = 0; i < 3; i++)
{
for (j = 0; j  < 3; j++)
{
if (*(*(mat + i) + j) < minimum)
minimum = * (* (mat + i) + j);
}
}

for (i = 0; i < 3; i ++)
{
for (j = 0; j < 3; j++)
{
if (i = = j)
* (*mat + i) + j) = minimum;
}
```

```
printf ("\n New matrix: \n");
for (i = 0; i < 3; i++)
{
for (j = 0; j < 3; j++)
printf ("%d", * (*(mat + i) + j));
printf ("\n");
}
return 0;
}
```

5. Write down advantages of pointer. Write a program using pointer to swap the value of two variables where the swapping operation is performed in function. [2071 Shrawan]

Ans: Refer to Theory portion on page no. 121

```
#include <stdio.h>
void swap (int *, int *);
int main ( )
{
int a, b;
printf ("Enter two numbers: \n");
scanf ("%d %d", &a, &b);
printf ("\n Before swapping, a = %d, b= %d," a, b);
swap (&a, &b);
printf ("\n After swapping, a = %d, b= %d", a, b);
return 0;
}

void swap (int *ptra, int *ptrb)
{
int swap;
swap = *ptra;
*ptra = *ptrb;
*ptrb = swap;
}
```

6. Compare array and pointer with exmple. [2070 Ashad]

Ans: Refer to theory portion on page no. 123 (One dimensional arrays and pointers)

7. **Write a program to read a string from user and use a user defined function to copy the content of the read string into another character array changing lower case letter, to upper if any use Pointer to process the string.** [2070 Ch

Ans:
```
#include <stdio.h>
void copy (char [ ], char [ ])
int main ( )
{
char str1 [100],  str2 [100];
printf ("Enter the string: \n");
scanf ("%s", str1);
copy (str1, str2);
printf ("\n New string: \n");
scanf ("%s", str2);
return 0;
}

void copy (char str1 [100], char str2 [00])
{
int i = 0, length = 0;
while (str1 [i]! = '\0')
length ++;
for (i = 0; i < length ; i++)
{
if (*(str1 + i) > 'a' && *(str1 + i) < 'z')
* (str1 + i) = *(str1 + i) - 32;
}
}
```

8. **A pointer variables is used to store address of some variable. However, we need to specify data type while declaring pointer variable. Why?** [2070 Chaitra]

Ans: A pointer variable points to the memory location of the variables. Its data type is also the data type of the variable to which it points. The data type is needed when dereferencing the pointer to access to its values.

Also, the increment and decrement on the pointer variable by 1 causes the increment or decrement on the number of bits of space occupied by the particular data type.

For E.g.
```
int *iptr;
char *cptr;
iptr ++;
cptr ++;
```
Here, memory location changes by 2 bytes for iptr and 1 byte for cptr.

Thus, we need to specify data type while declaring the variable.

9. **Briefly explain array of pointers. How are array and pointer related? Give example.** [2070 Chaitra]

Ans: When the multiple pointers of same type are stored in an array it is called as array of pointers.

Here, each elements contained by the array represent the pointers that contain the memory address of other variables.

Syntax of declaration:

*data_type * pointer_name* [size];

E.g. `float * ptr [100];`

Here, ptr is an array of 100 pointers each of which points to an floating point variable.

(Refer to theory portion on page no. 123)

10. **Explain call by reference. How are pointers used in call by reference?** [2069 Chaitra]

Ans: Refer to theory portion on page no 78

11. **Using pointer concept, write a program to count the number of characters and the number of words in a line of text entered by the user.** [2069 Chaitra]

Ans:
```
#include <stdio.h>
int main ( )
{
int i, ch = 0, space = 0;
char string [500];
printf ("Enter the text :\n");
gets (string);
while (*(string + i)! = '\0')
ch ++;
for (i = 0; i < ch; i++)
{
```

E. I Yr I Part)

Array

ıe collection of rela
f same type.

elements are stored
ıous memory locatio

elements are acces
r index number.

declaration
ıg operator is squa
"[ ]".

name points to
ment in that array.
ame is a pointer.

lements in structur
ıme size.

is no keyword
an array.

user define
data type.

name [size];

Solved Examination Questions

1. Explain the structures and nested structures? Create a structure to hold any complex number x + iy. Write a program that uses the structure to read two complex numbers and display the third complex number which is the multiplication of the entered complex numbers. [2074 Ashwin]

Ans: Refer to theory portion on page no. 137 & 141

```
#include <stdio.h>

structure complex
{
float x, y;
} C1, C2, p;

int main ( )
{
printf ("Enter x and y for first complex number: x +
iy");
scanf ("% f %f", &C1.x, &C1.y);
printf ("Enter x and y for second complex number:");
scanf ("%f %f", &C2.x, & C2.y);

p.x = C1.x * C2.x - C1.y * C2.y;
p.y = C1.x * C2.y + C1.y * C2.x;

printf ("Product: %.2f + %.2f i", p.x, p.y);

return 0;
}
```

2. What is nested structure? Write a program in C to read name, age and salary of 10 different employees as the three members of the structure named as "employee". Sort this data in salary basis using user defined function and display sorted data from main function. [2073 Chaitra]

Ans: If a structure contains another structure within it, it is called nesting of structure. Outer structure is called nesting structure and inner cture is known as nested structure.

```
#include <stdio.h>
struct employee
{
char name [40];
int age;
float salary;
}

void sort (struct employee e [ ]);
int main ( )
{
int i;
struct employee e[10];
printf ("Enter the records of 10 employee \n");
printf ("Name \t Age \t salary \n");

for (i = 0; i < 10; i ++)
{
scanf ("%s %d %f", e[i].name, & e[i].age,
   &e[i].salary);
printf ("\n");
}

sort (e);

printf ("\n Data according to salary of employee
\n");
printf ("\n Name \t Age \t Salary \n");
for (i = 0; i < 10; i++)
{
print f ("%s \t %d \t % . 2f", e [i] . name, e [i]
age, e [i] .. salary");
printf ("\n");
}
return 0;
}

void sort (struct employee e [ ])
{
Struct employee swap;
int i, j;
```



```
swap = e [0];
for (i = 0; i < 10; i++)
{
for (j = i + 1; j < 10; j++)
{
if (e [j] . salary < e [i] . salary)
{
swap = e [i];
e [i] = e [j];
e [j] = swap;
}
}
}
}
```

3. **Why should we prefer structure over array? Explain nested structure with example.** **[2073 Shrawan]**

Ans: Array is a collection of same type of elements but in many real life applications we may need to group different types of logically related data. Thus, to store the related fields of different data types, we cant use array. So, we should prefer structure over array.

(Refer to nested structure in theory portion on page no 141)

4. **Explain the need of structures. How can we create and use a structure within another** **[2072 Kartik]**

Ans: Structures are needed to represent the record that are used to group logically related different types of data.

For e.g. if we want to create a record of a person that contains name, age, sex, height and salary of a person, we can use structure as it is capable of storing heterogenous data.

(Refer to "Nested Structures" in theory portion on page no. 141)

5. **Explain dot and arrow operations for accessing the members of a structure.** **[2072 Kartik]**

Ans: **Dot operator:**

For accessing any member of a structure variable, we use dot (.) operator, which is also known as period or membership operator.

Syntax:

Structure_variable . member

Here, *Structure_variable* represents the variable of structure type and *member* represents the name of member of that structure.

Arrow operator:

Arrow operator is used for accessing member of structure using pointer variable.

Syntax:

ptr_variable –> member_name;

Where – > is known as arrow operator

Example:

```
struct student
{
char name [20];
int rollno;
int age;
}
struct student stu, *ptr;
```

There are two ways accessing the members of structures through structure pointer.

(* ptr) . name		ptr – > name
(* ptr) . rollno	OR	ptr – > rollno
(* pr) . age		ptr – > age

6. What is the principal difference between a structure and an array? [2072 Chaitra]

Ans: A structure can store related heterogenous data under common name but an array stores a related homogenous data under common name.

7. Write a program to read structure "college" having name, est Date and location where est Date is an another structure having day, month and year as members. Display the records of 10 colleges. [2072 Chaitra]

Ans:

```
#include <stdio.h>

int main ( )
{
struct college
{
char name [40];
char location [40];
struct estDate
```



```
{
int day;
int month;
int month;
int year;
} doe;
} col [10];
int i;

printf("Enter the records:");

for (i = 0; i < 10; i++)
{
printf("\n Record number: %d \n", i++);
printf("Name: \t");
scanf("%s", col[i].name);
printf("\t location: \t");
scanf("%s", col[i].location);
printf("Enter Date \n");
printf("Day \t month \t year");

scanf("%d %d %d", & col[i].doe.day, & col[i].doe
.month, & col[i].doe.year);
printf("\n");
}

printf("\n The record of colleges: \n");

printf("Name\t location\t Estd year\t month\t Day");

for (i = 0; i < 10; i ++)
{
printf("\n");
printf("%s \t %s \t %d \t %d \t %d", col[i].name,
col[i].location, col[i].doe.year, col[i].doe.month,
col[i].doe.day");
}
return 0;
}
```

8. Explain "arrays within structures" with programming example. [2071 Shra...

Ans: C permits array within structure. Inside an structure, we can ha... or more arrays.

Following example makes clear:

```
#include <stdio.h>

{ struct student
 char name [30];
int rollno;
int age;
int submarks [7];
}
int main ( )
{
int i, j;
struct student st [5];

for (i=0; i<5; i++)
{
printf("Enter data for student %d \n", i +1);
printf("Enter name:");
scanf("%s", st [i].name);
printf("\n Enter rollno");
scanf("%d", & st [i].rollno);
printf("Enter age:");
scanf("%d", & st [i].age);

for (j=0; j<4; j++)
{
printf ("Enter mark for subject %d \n", j + 1);
scanf ("%d", & st [i].submarks [j]);
}

printf ("\n Your records: \n");
for (i=0; i<3; i++)
{
printf (Data of student %d \n", i +1);
printf ("Name:%s \n", st[i].name);
```



```
printf ("Roll number: % d \n", st[i].age);
printf ("Subject marks: \n");
for (j=0; j<7; j++)
{
printf ("Subject:%d", j+1);
printf ("%d", st[i].submarks[j]);
printf ("\n");
}
return 0;
}
```

In above example, name is a character array and submarks is an integer array within the structure; student.

9. Write a program to understand how structure members are sent to a function. [2071 Shrawan]

Ans: Refer to theory portion on page no. 143

10. What do you mean by nested structures? Give suitable example. Write a program to read the heights of two students and display the difference between their heights. Use feet and inches as members of the structure to define height. [2070 Ashad]

Ans: Refer to theory portion on page no. 141

```
#include <stdio.h>
struct student
{
int feet;
int inches;
} st1, st2;

int main ( )
{
struct student difference;
printf ("Enter height of student1: \n");
printf ("feet = ");
scanf ("%d", st1.feet);
printf ("\n Inches =");
scanf ("%d", st1.inches);

printf ("Enter height of student2: \n");
```

```
    printf ("feet =");
    scanf ("%d", st2.feet);
    printf ("\n Inches =");
    scanf ("%d", st2.inches);

    if (st1.inches < st2.inches)
    {
    st1.inches = st1.inches + 12;
    st1.feet = st1.feet -1;
    }
    difference.inches = st1.inches - st2.inches;
    difference.feet = st1.feet - st2.feet;
    }
    if (difference.feet < 0)
    difference.feet = -1 * difference.feet;
    printf ("Difference in heights: \n");
    printf ("feet \t inches \n");
    printf ("%d \t %d", difference.feet, difference.
        inches);

    return 0;
    }
```

11. Write a program to display only those students information which are passed. Use separate function to check result of student. The information of students like Name, Roll No. Address and Marks are passed from the main functions and pass to functions using array type arguments. [2070 Chaitra]

Ans:

```
#include <stdio.h>

struct student
{
char name [40]
int roll;
char address [40];
int marks;
};

void pass (struct student s[ ], int);

int main ( )
```



```
{
struct student s[50];
int n, i;
printf ("Enter the no. of students:");
scanf ("%d", &n);
printf ("Enter the following data:\n");

for (i = 0; i < n; i ++)
{
printf ("Student %d\n", i +1)
printf ("Name \t Rollno \t Address \t marks");
scanf ("%s%d%s%d", s[i].name, &s[i].roll, s[i].
    address, & s[i] . marks);
printf ("\n");
}
pass (s, n);
return 0;
}

void pass (struct student s[ ], int n)
{
int i;
printf ("Name \t RollNo \t Address \t Marks: \n");
for (i = 0; i < n; i ++)
 {
 if (s[i] . marks > 40)
 {
 printf ("%s \t %d \t %s \t %d", s[i] . name, s[i] .
 roll, s[i] . address . s[i] . marks);
  }
  }
  }
```

12. Explain the use of typedef keyword in structures. [2070 Chaitra]

Ans: *typedef* keyword allows us to define a new name for an existing data type. It is used to give a new name to structure.

Example:

```
typedef struct book
 {
 char name [40];
```

```
    int pages;
    char author [40];
    };
```

Here, *typedef* is used to define the new datatype for the structure book.

typedef is optional keyword in structure. Thus, the above example can be modified as:

```
struct book
{
char name [40];
int pages;
char author [40];
};
```

13. Explain the need of nested structure. Write a C program to convert date in BS to date in AD using structure. Use the date difference of current date. [2070 Chaitra]

Ans: We need nested structures in such cases where the members have submembers. E.g. If we store records of a person, there may be members like name, age, height etc. here, height can have sub members : feet, inches etc.

```
#include <stdio.h>

struct date
{
int year, month, day;
};

int main ( )
{
struct date BS, AD;
printf ("enter year, month and day in BS:");
scanf ("%d %d %d", &BS.year, &BS.month, &BS.day);

AD.day = BS.day -17;
if (AD.day <= 0)
{
AD.day = AD.day + 30;
BS.month = BS.month -1;
}
```



```
AD.month = BS.month -8;
if (AD.month <=0)
{
AD.month = AD.month +12;
BS.year = BS.year -1;
}
AD.year = BS.year -56;
printf("Date in AD \n");
printf("year \t month \t Day \n");
printf("%d \t %d \t %d", AD.year, AD.month, AD.day);
return 0;
}
```

14. What is the advantage of using structure? Create an array of structure named employee with name and salary as structure member and the array of structure is passed to a function which sorts in ascending order on the basis of salary and display the sorted array from main. [2069 Chaitra]

Ans: Refer to 2070 Chaitra for first question on page no 158

Refer to 2073 Chaitra for second question on page no 149

15. Why structure variable differs from array? Write a program to input name, post and salary of ten employees from main function and pass to structure type user defined function. (arguments of this function should also be a structure type). This function returns structure variable which keeps the records of only those employees whose salary is greater than 10,000. This modified record is also displayed from main ( ) function. [2068 Chaitra]

Ans: Refer to theory portion on page no. 148

```
#include <stdio.h>

struct employees
{
char name [40];
char post [40];
float salary;
};

struct employees *modify (struct employees [ ],
int*);
```

Solved Examination Questions

1. What are different input/output functions used with data files in C? Explain with syntax and examples. [2074 Ashwin]

Ans: Refer to theory portion on page no 167

2. Write a program in C to read integers from user until user says 'no'. After reading the data write all the odd numbers to the file named odd.txt and all the even number to file named even.txt. [2074 Ashwin, 2069 Chaitra]

Ans:
```
#include <stdio.h>
#include <stlib.h>
#include <string.h>

int main ( )
{
FILE *fptr1, *fptr2;
int n;
char chk [10];

fptr1 = fopen ("ODD", "w");
fptr2 = fopen ("EVEN", "w");

if (fptr1 = = fptr2 = = NULL)
{
printf ("Sorry!! Files can't be opened.");
exit (12);
}

do
{
printf ("Enter a number: \n");
scanf ("%d", &n);
if (n %2 = = 0)
fprintf (fptr2, "%d", n);
else
fprintf (fptr1, "%d", n);
printf ("Do you want to add more? Yes/No");
scanf ("%s", chk);
```



```
} while (strcmp (chk, no) ! = 0));
fcloseall ( );
return 0;
}
```

3. Write a program in C-programming language to store in it for maximum 10000 data of ITEM_NAME, NUMBER, PRICE, QUANTITY. Extend the program to read this data from the above given filename and display the inventory table with the value of each item. [2073 Chaitra]

Ans:
```
#include <stdio.h>
#include <stlib.n>

int main ( )
{
FILE *fptr;
int i, n = 0;
char ch;
fptr = fopen ("INVENTORY", 'w+");
char ITEM_NAME [40];
int NUMBER;
float PRICE;
int QUANTITY;
printf ("Enter following data: \n");
do
{
printf ("\n Item name:");
scanf ("%s", ITEM_NAME);
printf ("\n Number:");
scanf ("%d", & NUMBER);
printf ("\n Enter price:");
scanf ("%f", & PRICE);
printf ("\n Enter Quantity:);
scanf ("%d", & QUANTITY");
n++;
printf ("Do you want to enter more? Y/N");
scanf ("%c", &ch);

fprintf (fptr, "%s %d %.2f %d", ITEM.NAME, NUMBER,
PRICE, QUANTITY);
```

```
printf ("\n");
} while (ch = = 'Y' && n < = 1000);
rewind (fptr);
printf ("Item name \t number \t price \t Quantity
\n");
for (i = 0; i < n; i++)
{
while (fscanf (fptr, "%s %s %d %f %d", ITEM_NAME, &
NUMBER, & PRICE, & QUANTITY) ! = EOF)
printf ("%s \t %d \t % 2f \t %d", ITEM_NAME, NUMBER,
PRICE, QUANTITY);
}
fclose (fptr);
return 0;
}
```

4. Write a program to read name and roll number of 48 students from user and store them in a file. If the file already contains data, your program should add new data at the end of the file. [2073 Shrawan]

Ans:

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

int main ( )
{
FILE *fptr;
int i;
char name [40];
int roll;
fptr = fopen ("student", "a");
printf ("\n Enter records: \n");
for (i = 0; i < 48; i++);
{
scanf ("%s %d", name, &roll);
fprintf (fptr, "%s %d", name, roll);
}
fclose (fptr);
return 0;
}
```


5. List different types of standard I/O used in C. Write a program to write name, roll no and age of five students into a disk file name "STUDENT.DAT". [2072 Kartik]

Ans: Different types of standard I/ used in C are:

(I) Character I/O - fgetc ( ), fputc ( )
(II) String I/O - fgets ( ), fputs ( )
(III) Formatted I/O - fscanf ( ), fprintf ( )
(IV) Record I/O - fread ( ), fwrite ( )

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

in main ( )
{
FILE *fptr;
int i;
char name [40];
int roll;
int age;
fptr = fopen ("STUDENT.DAT", "w");
if (fptr == NULL)
{
printf ("File can't be opened");
exit (1);
}

printf ("Enter the records :\n");

for (i=0; i < 5; i++)
{
printf ("Student %d", i+1);
printf ("Name:");
scanf ("%s", name);
printf ("\n Roll no:");
scanf ("%d", &roll);
printf ("\n Age:");
scanf ("%d", &age);
printf ("\n");

fprintf (fptr, "%s %d %d", name, roll, age);
```

```
return 0;
}
fclose (ptr);
return 0;
}
```

6. What is a data file in C? What are the modes in file handeling? Explain briefly. [2072 Chaitra]

Ans: Refer to Theory portion on page no.163 & 165

7. Write a program to read the information of a file named "data.txt" and write its contents to another file "record.txt". [2072 Chaitra]

Ans:

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

int main ( )
{
FILE *fptr1, *fptr2;
char ch;
fptr1 = fopen ("data.txt", "r");
fptr2 = fopen ("record.txt", "w");
if (fptr1 == NULL)
{
printf ("File named data.txt not found");
exit (1);
}
if (fptr2 = = NULL)
{
printf ("File named record.txt can't be created");
exit(1);
}

while ((ch = fgetc (fptr1))!= EOF)
{
fputc (ch, fptr2);
}
printf ("Data copied successfully.");
fclose (fptr2);
fclose (fptr1);
return 0;
}
```


5. *Write a program to store employee details in a text file. Read data from the text file, sort them in ascending order of salary and store the sorted record to the binary file. Display the details and rank of employee given by user.* [2071 Shrawan]

Ans:

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

struct employee
{
char name [40];
int rank;
float salary;
};

int main ( )
{
int n, i, j;
FILE *tptr, *bptr;
struct employee *e, temp;

tptr = fopen ("Employee.txt", "r");
printf ("Enter no. of employee");
scanf ("%d", &n);
if (tptr = = NULL)
{
printf ("File can't be opened");
exit (1);
}

printf ("Enter details of :");

for (i = 0; i < n; i++)
{
printf ("Employee %d \n", i+1);
printf ("\n Name:");
scanf ("%s", (e+i) -> name);
printf ("\n Rank:");
scanf (" %d", & (e + i) -> rank);
printf ("\n salary:");
scanf ("%f", & (e + i) -> salary);
fwrite (&e, size of (e), 1, tptr);
}
```

```
fclose (tptr);
bptr = fopen ("employee.txt", "r+b");
if (bptr = = NULL)
{
printf ("Error in opening file");
exit (1);
}
for (i = 0; i < n; i++)
fread (&e, size of (e), 1, bptr);
for (i = 0; i <n-1; i++)
{
for (j = i + 1; j <n; j++)
{
if ((e + i) -> salary < (e + j) -> salary)
{
temp = *(e + i);
*(e + i) = *(e + j);
*(e + j) = temp;
}
}
}
printf ("The record in ascending order: \n");
printf ("Name \t Salary \t Rank \n");
for (i = 0; i < n; i++)
{
fwrite (&e, size of (e), 1, bptr);
printf ("%s %.2f %d", (e+i) -> name, (e+i) ->
salary, (e + 1) -> rank);
}
fclose (bptr);
return 0;
}
```

9. Write a program to read the details of book authors and write it to a file until the user confirms to end. Then read and display the n^{th} record in the file, where n is read from user. The data for authors must be represented by structures that contain name, nationality and number of books published. [2070 Ashad]

Ans:

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



```
struct author
{
char name [40];
char nationality [20];
int number;
};

int main ( )
{
struct author aut;
int n;
char ch;
FILE *fptr;
fptr = fopen ("Author.rec", "wb");

if (fptr = = NULL)
{
printf ("File can't be opened!!");
exit (1);
}
printf ("Enter records: \n");
do
{
printf ("\n Name");
scanf ("%s", aut.name);
printf ("\n Nationality");
scanf ("%s", aut.nationality);
printf ("\n No. of books");
scanf ("%d", &aut.number);
fwrite (&aut, size of(aut), 1, fptr);
printf ("Do you want to add more recoreds? Y/N");
scanf ("%c", &ch);
}while (ch = = 'Y' || ch = = 'y');

fclose (fptr);
fptr = fopen ("Author.rec", "rb");
printf ("enter the position of record: \n");
scanf ("%d", &n);
fseek (fptr, (n - 1) * size of (aut), 0);
fread (&aut, size of (aut), 1, fptr);
```

```
printf ("Your details: \n");
printf ("Name \t Nationality \t No. of books: \n");
printf ("%s \t %s \t %d", aut.name, aut.nationality,
aut.number);
fclose (fptr);

return 0;
}
```

10. Define opening and closing a file with suitable examples.

Ans: Refer to theroy portion on page no 167

11. Write a program to display the records in sorted order. Sorting is performed in ascending order with respect to name using data file concept. [2070 Chaitra]

Ans:
```
#include <stdio.h>
#include <stlib.h>
#include <string.h>

int main ( )
{
FILE *fptr;
char str [25] [30], temp [30];
int i, j, n;

fptr = fopen ("Name.txt", "r");
if (fptr = = NULL)
{
printf ("File can't be opened.");
exit (1);
}

printf ("Enter the number of reco
scanf ("%d", &n);

for (i = 0; i < n ; i++)
fscanf (fptr, "%s", str

for (i = 0; i < n -
{
```



```
for (j = i + 1; j < n; j++)
{
if (strcmp (str [j], str [i] < 0)
{
strcpy (temp, str [j]);
strcpy (str [j], str [i];
strcpy (str [i], temp);
}
}
}
printf ("Data in ascending order \n");

for (i = 0; i < n; i++)
printf ("\n %s", str [i]);

fclose (fptr);
return 0;
}
```

12. Differentiate between text file and binary file. [2069 Chaitra]

Ans: Refer to theory portion on page no 165

13. Write a program to input and save record like name, roll, address and obtained makrs of 48 students in a binary file and search and display record of a student whose obtained mark is highest. The information should be organized in a structure. [2068 Chaitra]

Ans:
```
#include <stdio.h>
#include <stlib.h>

struct record
{
char name [40];
int roll;
char address [40];
int marks;
} st;

int main ( )
{
struct record *st, temp;
```

The most common format code places are:

A – for string

I – integer

D – Double precision numbers, exponent notation

E – Real numbers, exponent notation

F – Real numbers, fixed point format

0 – (horizontal skip space)

/ – Vertical skip new line

Solved Examination Questions

1. What are the characteristics of FORTRAN programming? Write a program in FORTRAN to calculate the value of π by evaluating the following formula for the first 25 terms. [2074 Ashwin]

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots\right)$$

Ans: Characteristics of FORTRAN programming are:

- It is mostly used for numeric operations and calculations.
- Has a lot of existing code and rich library.
- More efficient mathematics.
- Programs are portable and machine independent.
- Execution is faster and more efficient.
- Has poor string handling functions.

```
Program calculate
Integer i, j, sign
real, pi = 1
j = 1
sign = 1
do i = 1, 25

pi = pi + (1/j) * sign
j = (j * 2 + 1)
sign = sign * ( - 1)
end do
pi = pi * 4
write (*,*) "π = ", pi
end
```

2. Explain different types of GOTO statements in FORTRAN programming with suitable. Write a program to read n from the user and display the sum of following series till nth terms.
1 + (1 + 2) + (1 + 2 + 3) + (1 + 2 + 3 + 4) + ... (....n) [2072 Kartik]

Ans: There are two types of GOTO statements in FORTRAN prorgramming

(i) Unconditional GOTO

(ii) Computed GOTO

(i) Unconditional GOTO:

When a program encounters an unconditional GOTO statement, control immediately passes to the label.

Syntax:

GOTO label

.....

.....

label statement

The following program shows the use of unconditional GOTO statement.

```
PROGRAM use of GO TO
integer n
write (*, *), 'Enter a number'
read (*, *), n
GOTO 7
6 write (*, *), n + 1
7 write (*, *), n
if (n . GT . 4) then GOTO 6
end
```

(ii) Computed GOTO

When a program encounters computed GOTO statement, control is transferred to one of the several statements depending upon the value of integer in expression.

Syntax:

GO TO (n_1, n_2, n_3 ..... nk), j

The following program shows the use of computed GO TO statement

```
Program computed goto
integer j
write (*,*) 'Enter a number from 1 to 3'
read (*,*) j
GOTO (10, 20, 30) j
10 write (*,*) 'you have chose 1'
20 write (*,*) 'you have chose 2'
30 write (*,*) 'you have chose 3'
end
```


Program to display sum of series:
1 + (1+2) + (1+2+3) + (1+2+3+4) + ... n

```
program sum
integer s, i, j, n
write (*,*) 'enter value of n:'
read (*,*) n
do i = 1, n
do j = 1, i
s = s + j
end do
end do
write (*,*) 'sum of series is:' s
end
```

3. Explain with suitable example to show how an implied Do loop works in FORTRAN. [2072 Kartik]

Ans: Implied do loop is the implicit form of do loop in FORTRAN.

Syntax:

(array-element, counter-variable = initial value, final value, step-size)

Program to show a sum of 2×2 matrix using implied Do loop

```
Program sum
integer num1 (2, 2), num2 (2, 2), sum (2, 2), i, j
write (*,*) 'Enter elements of first matrix
do i = 1, 2
read (*,*); (num1 (i, j), j = 1, 2)
end do
write (*,*) 'Enter elements of second matrix'
do i = 1, 2
read (*,*), (num2 (i, j), j = 1, 2)
end do
do i = 1, 2
do j = 1, 2
sum (i, j) = num1 (i, j) + num2 (i, j)
end do
end do
write (*,*), 'sum is'
```

```
do i = 1, 2
write (*,*), (aum(i, j), j=1, 5)
stop
end
```

4. What is the structure of FORTRAN program? [2073 Shrawan]

Ans: A FORTRAN program consists of a main program, possibly followed by one or more sub programs.

The basic structure is given in following scheme:

Program name
declaration of variables
initialization of variables
statements
end

Lets explain the structure FORTRAN program with following example.

```
program integers
integer i
write (*,*) 'Integers from 1 to 50:'
do i = 1, 50
write (*,*), i
end do
end
```

(i) Program name:

It is the first line of program that gives name for it. Here, 'integers' give the name for program.

(ii) Declaration section:

Here, variables necessary for program are declared. Here, 'i' is the variable declared as integer in the program.

(iii) Initialization of variables:

If the value for variable is to be predefined. It is done in this section.

(iv) Statements:

It is the body section of the program where all the executable FORTRAN statements are written.

(v) end:

It is the last statement of FORTRAN used to terminate the program.

5. Write a FORTRAN program to read n numbers and display the largest among them.

Ans:

```
Program largest
integer num (100), n, large, i
write (*,*), 'Enter the number of terms.'
read (*,*), n
large = num(1)
do i = 1, n
if (num (i) . GT . large)
large = num (i)
end do
write (*,*), 'The largest number is', large
stop
end
```

6. Compare DO and implied DO statement in FORTRAN. [2072 Chaitra]

Ans: DO loop is the count controlled loop which allows a block of statements to be executed repeatedly for predetermined number of times. It uses loop control variable that have initial and final values.

The syntax is given by,

DO label, loop control_variable = initial value, final value, step_size
statement - 1
statement - 2
.....
.....
statement n

Label CONTINUE

Similarly,

implied DO statement is the implicit (shortened forms of do loop. It is mostly used when large number of variables are to be initialized or displayed.

In a single DO statement, various statements can be written within, which is not possible in implied DO statement.

syntax for implied do statement:

array_element, counter_variable = initial value, final value, step_size.

7. Write a FORTRAN program to add and subtract two matrices and display the results in matrix form. [2072 Chaitra]

Ans: Program add and subtract

```
integer mat1(10, 10), mat2(10, 10), add (10, 10) sub
(10, 10)
integer i, j, m, n
write (*,*); 'enter the order of matrix less than
10'
read (*,*), m, n
write (*,*), 'Enter the elements of 1st matrix:'
do i = 1, m
read (*,*), (mat1(i, j), j = 1, n)
end do
write (*,*), 'Enter 2nd matrix elements'
do i = 1, m
read (*,*), (mat2(i, j), j = 1, n)
end do
do i = 1, m
do j = 1, n
sub (i, j) = mat1(i, j) - mat2 (i, j)
add (i, j) = mat 1(i, j) + mat2 (i, j)
end do
end do

write (*,*), 'The sum is:'
do i = 1, m
write (*,*) (add (i, j), j = 1, n)
end do

write (*,*), 'The difference is:'
do i = 1, m
write (*,*) (sub(i, j), j = 1, n)
end do
end
```

8. What do you mean by formatted and unformatted input/output statements in FORTRAN and also give suitable example which explain the concept of formatted i/o. [2071 Shrawan]

Ans: Refer to theory portion on page no 198

9. Write a program to convert a binary number to a decimal number using FORTRAN programming language. [2071 Shrawan]

Ans: Program conversion

```
integer num, bin_val, dec_val, base, rem
base = 1
dec_val = 0

write (*, *) 'Enter a binary number '
read (*, *), num
bin_val = num

do while (num . GT . 0)
rem = num mod 10
dec_val = dec_val + rem * base
num = num/10
base = base * 2
end do

write (*, *) 'Binary number:', bin_val
write (*, *) 'Decimal-value:', dec_val
end
```

10. Explain the FORTRAN structure. What are data types in fortran? [2070 Ashad]

Ans: First part: Refer to 2073 shrawan on page no. 202

The types of data defined for each variables are called data types.

The data types used in fortran are:

(i) Integer
(ii) Real
(iii) Double precision
(iv) Complex
(v) Logical
(vi) Character

11. Write a program in FORTRAN to solve quadratic equation and display roots in proper format. [2070 Ashad]

Ans: Program roots

```
integer a, b, c, sqt
real root1, root2
```

```
write (*, *), 'Enter values a, b, c'
read (*, *), a, b, c
sqt = b * b - 4 * a * c
if (sqt . EQ . 0) then
write (*, *) 'Root is :' (-b)/(2.* a)

else if (sqt . GT . 0) then
root1 = (-b)/(2 * a) + sqt ** (1/2)
root2 = (-b)/(2 * a) - sqt ** (1/2)
write (*, *), 'Roots are: 'root1, root2
else
write (*, *), 'Roots are:' ((-b)/(2 * a) + (-sqt) **
(1/2) 'i' 'and', ((-b)/2*a) - (-sqt) ** (1/2) 'i'
end
```

12. Compare arithmetic and logical if statements in FORTRAN with suitable example. [2070 Chaitra]

Ans: The arithmetic if statement branches to one of the three specified statements depending on the value of an arithmetic expression.

Syntax: *if (expression) S1, S2, S3*

The if statement transfers control to the first second or third label if the value of arithmetic expression is less than zero, equal to zero or greater than zero respectively.

E.g. P = 8

if (P), 10, 20, 30

Since P > 0, control is transferred to statement label 30

The logical IF statement executes one single statement, or doesn't execute it depending on the value of logical expression.

Syntax: *if (expression) statement*

The logical if statement evaluates a logical expression and executes the specified statement if the value of logical expression is true and doesn't execute if logical expression is false.

E.g. N = 8

If (N mod 2 = 0) write (*, *), N 'is even'

13. Write a FORTRAN program to read m * n matrix, transpose it and display both the matrices. [2070 chaitra]

Ans:
```
Program transpose
integer mat (10,10), trans (10,10)
integer i, j, m, n
write (*,*), 'Enter order of matrix less than 10x10'
read (*,*), m, n
write (*,*), 'Enter elements of matrix'
    do i = 1, m
    read (*,*) (mat(i,j), j = 1, n)
    do i = 1, m
    do j = 1, n
    trans (i, j) = mat (j,i)
    end do
    end do
write (*,*) 'Input matrix:'
do i = 1, m
write (*,*) (mat (i, j), j = 1, n)
end do
write (*,*) 'Transpose matrix:'
do i = 1, m
write (*,*) (trans (i, j), j = 1, n)
end do
end
```

14. Differentiate 'logical if' with 'Arithmetic if' in FORTRAN with suitable example. [2069 Chaitra]

Ans: Refer to 2070 Chaitra on page no. 206.

15. Compare "Computed goto" statement [FORTRAN] and "switch" (c-language). Write a program to read a day number and display whether it is Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday using both concept. [2070 Chaitra]

Ans: When a FORTRAN program encounters computed GOTO statement control is transferred to one of the several statements depending on the value of integer in the expression.

Syntax: GOTO (*n1, n2, n3, + ... nk), j*

The control passes to n1 if j = 1
n2 if j = 2
.....
nk if j = k
and so on.