Q9(b). Give an example of :
(i) Switch statement
(ii) Conditional operators
(iii) Nesting of Loops

Ans(i). Switch Statement :

If a programmer has to choose one block of statement among many alternatives, nested if_else can be used, but this makes programming logic complex. This type of problem can be handled using switch statement.

Syntax :

```
switch (n)
{
    case constant1 :
    codes to be executed if n = constant1;
    break;
    case constant2;
    codes to be executed if n = constant2;
    break;
    :
    default :
    codes to be executed if 'n' doesn't match to any cases :
}
```

Example :

```
void main( )
{
    int a;
    printf("enter the choice 1/2 :");
    scanf("%d", &a);
    switch(a)
    {
        case1 : printf("good morning");
        break;
        case2 : printf("good night");
        break;
```

```
        default : printf("bye");
    }
}
```

Output :

enter the choice 1/2 : 1
good morning.

(ii) Conditional Operators (?:) :

Conditional operators are used in decision making in C programming; i.e, executes different statements according to test condition whether it is either true or false.

Syntax of conditional operators :

Conditional_expression? expression1 : expression2

If the test condition is true, expression1 is returned and if false expression2 is returned.

Example of conditional operator :

```
#include <stdio.h>
void main( )
{   char feb;
    int days;
    printf("Enter 1 if the year is leap year other-
    wise enter 0 :");
    scanf("%c", &feb);
    days = (feb = = '1')? 29 : 28;
    printf("number of days in February = %d",
    days);
}
```

Output :

Enter 1 if the year is leap year otherwise enter 0 : 1

Number of days in February = 29

(iii) Nesting of Loops :

In many cases we may use loop statement inside another looping statement. This type of looping is called nested loop. In nested loop, the inner loop is executed first & then outer.

Syntax :

The syntax for a nested loop statement in C is as follows :

```
for(init; condition; increment)
{
    for(init; condition; increment)
    {
        statement(s);
    }
    statement(s);
}
```

Example :

The following program uses a nested loop to find prime numbers from 2 to 20.

```
#include <stdio.h>
void main( )
{
    int i, j;
    for(i = 2; i < 20; i++)
    {
        for(j = 2; j < = (i/j); j++)
        if(! (i % j)) break;
        if(j > (i/j)) printf("%d is prime \n", i);
    }
}
```

Output :

2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime

Q10. What is the file pointer? Write a program to copy the contents of one file into another.

Ans. File pointer :

It is a struct datatype which has been defined in standard library stdio.h. This datatype points to a stream or a null value. It has been defined in stdio.h as

```
typedef struct
{
    short level;
    unsigned flags;
    char fd;
    unsigned char hold;
    short bsize;
    unsigned char *buffer, *curp;
    unsigned istemp;
    short token;
}
FILE;
```

Program to copy contents of one file into another :

```
#include <stdlib.h>
void main( )
{
    FILE *fptr1, *fptr2;
```

```
    char filename [100], c;
    printf("Enter the filename to open for reading
    \n");
    scanf("%s", filename);
                        // open one file for reading
    fptr1 = fopen (filename, "r");
    if(fptr1 = = NULL)
    {
        printf("cannot open file % s \n", filename);
        exit (0);
    }
    printf("Enter the filename to open for writing
    \n");
    scanf("%s", filename);
                // open another file for writing
    fptr2 = fopen (filename, "w");
    if(fptr2 = = NULL)
    {
        printf("cannot open file %s \n", filename);
        exit (0);
    }
    // Read contents from file
    c = fgetc (fptr1);
    while (c! = EOF)
    {
        fputc (c, fptr2);
        c = fgetc (fptr);
    }
    printf("\n contents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

Output :

```
Enter the filename to open for reading
a.txt
Enter the filename to open for writing
b.txt
contents copied to b.txt
```

Q11(a). Write 'C' function void it pa(int n, char s[[]), which converts integer n into string S.

Ans. C program to convert an integer to string:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
void it pa (int n, char s[])
{
    int i, rem, len = 0, num;
    num = n;
    while(num! = 0)
    {   len++;
        num/ = 10;
    }
    for(i = 0; i < len; i++)
    {   rem = n% 10;
        n = n / 10;
        s (len – (i + 1)) = rem + '0';
    }
    s[len] = '\0';
}
void main( )
{   char str [0];
    int num, result;
    printf("enter a number :");
    scanf("%d", &num);
    itpa (str, num);
    printf("number converted to string : % s\n",
    str);
}
```

Output :

```
enter a number : 12345
number converted to string : 12345
```

Q11(b). What are jumping statements? Explain the difference between break and continue statements?

Ans. Jumping Statements :

C language provides multiple statements through which we can transfer the control anywhere in the program. There are basically 3 jumping statements:

(a) break jumping statements
(b) continue jumping statements
(c) goto jumping statements

(a) Break Statement :

In C programming, break jumping statement is used in terminating the loop immediately after it is encountered. The break statement is used with conditional if statement.

Syntax : break;

This jumping statement always used with the control structure like switch case, while, do while, for loop etc.

Example:

```
void main( )
{
    float num, avg, sum;
    int i, n;
    printf("maximum number of inputs \n");
    scanf("%d", &n);
```

```
    for(i = 1; i <= n; i++)
    {
        printf("enter n%d :" i);
        scanf("%f", &num);
        if(num < 0 – 0)
        break;
        sum = sum + num;
    }
    avg = sum / (i – 1);
    printf("average = %2f", avg);
}
```

Output :

```
maximum number of inputs
4
enter n1 : 1.5
enter n2 : 12.5
enter n3 : 7.2
enter n4 : 1
Average = 7.07
```

(b) Continue Jumping Statement :

It is sometimes desirable to skip some statements inside the loop. In such cases, continue statements are used.

Syntax : Continue;

This jumping statement is always used with control structures like switch case, while, do while, for loop etc.

Example Program :

```
void main( )
{
    int i, product, num;
    for (i = 1, product = 1; i <= 4; i++)
    {
        printf("enter num %d", i);
        scanf("%d", & num);
        if(num = = 0)
        continue;
        product * = num;
    }
    printf("product = %d", product);
}
```

Output :

```
Enter num1 : 3
Enter num2 : 0
Enter num3 : 5
Enter num4 : 2
Product = 30
```

(c) goto jumping statement :

It is used to transfer the control from the current location to anywhere in the program.

To do this, we have to specify a label with goto & the control will transfer to the location where the label is specified.

Example,

```
void main( )
{
    int i, n;
    float s;
    start : printf ("enter a number");
    scanf("%d", &n);
    s = sqrt (n);
    if(n <= 0)
    goto start;
    printf("%f", &);
}
```

Q12(a). What is a function and function Prototyping? Explain the different Methods of Passing parameters to a function by taking suitable C language program?

Ans. Function : A function is a self-contained program segment that carries out specific, well defined task. Every C program consists of one or more functions but it cannot execute without main function.

Function basically provides the modular approach as it is very difficult to trace, debug the program of huge lengths for mistakes or execution. Therefore we divide it on the basis of task and frame functions.

Syntax :

```
return_type function_name (parameter_list)
        //prototype
return_type function_name (parameter_list)
{       // function definition
  –     // function body
  –
  –
}
```

Prototype need not contain variable name and is terminated by semicolon. It is written in the program before the function is used and defined. It is mainly written when we define the function after using it in the sequence of program, so as to indicate presence of function by that name.

Function definition includes three things :

1. Return-type which indicates the type of data to be returned by the function.
2. Function-name which is an identifier. It is