RECURSION

A function calling itself again and again to compute a value is referred to recursive function or recursion. Normally a function is called by the main program or by some other function but in recursion the same function is called by itself repeatedly. Consider the following example to find factorial of an integer by recursion.

```
int fact(int k)
{
   if (k == 0)
      return(1);
   else
      return(k * fact(k - 1));
```

When this function is called, the value of k is supplied by the main program to find its factorial. Let k = 4. Then the sequence of the function called by itself is shown below to return the result.

After the first call which is from the main program.

```
return(4 * fact(4-1));
```

calls the function with k = 3

After the second call which is from the function

```
return(4 * 3 * fact(3-1));
```

calls the function with k = 2

After the third call which is from the function

```
return(4 * 3 * 2 * fact(2-1));
```

calls the function with k = 1

After the fourth call from the function

```
return(4 * 3 * 2 * 1);        /* result 24 is returned */
```

This will return the result to the main program and the repeated execution of the function is terminated.

Recursion is a programming technique based on functions calling themselves repeatedly.

Use of Recursive Function

1. Recursive functions are written with less number of statements compared functions.
2. Recursion is effective where terms are generated successively to compute a value.
3. Recursion is useful for branching processes. Recursion helps to create short code that would otherwise be impossible.

Example 11

Write a recursive function to find the factorial of a given integer. Use it to find

$$nCr = \frac{n!}{r!(n-r)!}$$

Solution

This program is similar to Example 1 except that the function is written as a recursive function.

```
#include <stdio.h>
#include <conio.h>
main()
{ int fact(int k);
   /* function prototype declaration */
  int n,r,ncr;
  clrscr();
  printf("\n Enter values to n and r : ");
```

```
    scanf("%d %d",&n,&r);
    ncr = fact(n)/(fact(r)*fact(n-r));
    printf("\n Value of nCr = %d",ncr);
    getch();
  }
 /* recursive function to find factorial */
 int fact(int k)
 {  if(k == 1)
      return(1);
    else
      return(k*fact(k-1));
 }
```

Also write the

Example 13

Write a recursive function to display the first n terms of the Fibonacci series.

0 1 1 2 3 5 8 13 . . . .

Also write the main program.

Solution

This program is similar to Example 14 of Chapter 6 except that a recursive function is written to generate and print the terms of the series.

```
#include <stdio.h>
#include <conio.h>
int n,count,t1,t2,t3;   /* Global declaration */
int fibo(int a,int b);   /* function prototype */
main()
{ clrscr();
  printf("\n How many terms to be printed ? ");
  scanf("%d",&n);
```

```
    t1 = 0;
    t2 = 1;
    printf("\n The first %d terms in Fibonacci series are \n",n);
    printf("%5d %5d",t1,t2);
    count = 2;  /* represents number of terms */
    fibo(t1,t2);
    getch();
  }
  /* recursive function to print the terms in series */
  int fibo(int t1,int t2)
  { if(count >= n)
      return;
    else
      { t3 = t1 + t2;
        printf("%5d",t3);
        count++;
        t1 = t2;
        t2 = t3;
        fibo(t1,t2);
      }
  }
```

numbers in a ... ending order. The values of 1,
N and set of N numbers are to be read by the main program.

SHORT QUESTIONS AND ANSWERS

1. A function may be placed either _____ or _____ the main function.

before, after

2. The four available storage classes are _____, _____, _____ and _____.

auto, static, extern, register

3. What is meant by recursion?

Recursion means calling a function itself again and again to compute a value.

4. What is meant by prototyping?

Prototyping means declaration of a function name along with the type and optional dummy arguments at the beginning of the calling program.

Example:

```
int fact(int k);
```

5. Write a function in C to calculate n!

```
int fact(int k)
{
   int p = 1, i;
   for(i = 1; i <= k; i++)
      p = p * i;
   return(p);
}
```

6. What is a static variable? When should it be used?

   A variable declared in static storage class is referred as static variable. It is useful to initialize the variable at the time of compilation and its latest value is retained throughout the program.

   Example:

   ```
   static int m, n, s = 0;
   ```

7. What is a register variable in C?

   A variable declared in register storage class is referred as register variable. These variable values are stored in the CPU memory register.

   Example:

   ```
   register int n;
   ```

8. Explain the concept and use of type **void.**

   A function performing a task rather than computing a value can be declared as a **void** function. It can be used to compute many values and transfer them to the calling program through global or external declarations.

9. A function is _____ taken for reference by the appearance of its name in a program.

   always

10. Distinguish between **auto** and **register** storage classes.

    An **auto** storage class uses RAM to store the variable values, and a **register** storage class uses the CPU memory register. Any number of variables is declared using **auto** storage class but only a few variables are declared using **register** storage class.

11. Explain the scope of a variable in C.

    Scope of a variable refers to the activeness of a variable in a program. It depends on the storage class and block in which it is declared. In the default auto storage class, the variable is active in the block in which it is declared.

12. Write a simple program with actual and formal arguments.

```
main()
{
    /* prototype declaration */
    int big(int x, int y, int z);        /* x, y and z are dummy arguments */
    int t1, t2, t3, a1, a2, a3;
    scanf("%d %d %d", &t1, &t2, &t3);
    scanf("%d %d %d", &a1, &a2, &a3);
    total = big(t1, t2, t3) + big(a1, a2, a3); /* Actual arguments */
    printf("\n Total marks = %d", total);
}
/* function subprogram */
int big(int a, int b, int c)            /* a, b and c are formal arguments */
```

```
{
    if(a > b)
        if(a > c)
            return(a);
        else
            return(c);
    else
        if(b > c);
            return(b);
        else
            return(c);
}
```

13. Distinguish between actual and formal arguments.

Actual arguments are variables whose values are supplied to the function in any function call. Formal arguments are variables used to receive the values of actual arguments from the calling program.

14. What is meant by global variable?

A global variable is common to the main program and function subprogram. Global variables are declared outside the **main()**, or they may be declared using **extern** storage class.

15. The two ways in which values can be passed to a function are using _____ and _____.

arguments, global declaration

16. What is the difference between function declaration and function definition in C?

A function declaration is used to declare the data type of the value returned by the function and the types of arguments expected.

A function definition has a list of statements used to compute a value or to perform a task.

17. Storage class is concerned with _____ and _____ of the variable.

storage location (RAM or CPU register), scope

18. Differentiate built-in functions and user-defined functions.

Built-in functions are used to perform standard operations such as finding the square root of a number, absolute value and so on. These are available along with the C compiler and are included in a program using the header files **math.h, string.h** and so on.

User-defined functions are written by the user or programmer to compute a value or perform a task. It contains a statement block which is executed during the runtime whenever it is called by the main program.

19. What are the major components of a function definition?

A function definition has the function declaration and a block of statements. It has the following form.

```
type fn_name(type argu1, type argu2..........type argu n)
{
        < local declaration>
        <statement block>
        return(variable or expression);
}
```

where type is the data type of value returned by the function and arguments expected.

argu1, argu2...argu n are arguments which are variables that receive values from the calling program.

fn_name is the name of the function which the function is called by the calling program.

20. When is the execution of a function terminated?

The execution of a function is terminated when a return statement is executed or when the last statement in a statement block of a function is executed.

21. What is meant by modular programming?

Modular programming refers to the smaller modules which are written as functions. A large program is normally divided into many modules or functions to compute a value or to perform a task which is easier to write and debug.