B.Tech I Year

Regular Course Handbook

Subject Name: Programming for Problem Solving (Unit-3)

BCS101 / BCS201: PROGRAMMING FOR PROBLEM SOLVING

Content	Contact Hours
Unit -1:	8
Introduction to Components of a Computer System: Memory, Processor, I/O Devices, Storage, Operating System, Concept of Assembler, Compiler, Interpreter, Loader and Linker. **Idea of Algorithm:** Representation of Algorithm, Flowchart, Pseudo Code with Examples, From Algorithms to Programs, Source Code. **Programming Basics:** Structure of C Program, Writing and Executing the First C Program, Syntax and Logical Errors in Compilation, Object and Executable Code. Components of C Language. Standard I/O in C , Fundamental Data types, Variables and Memory Locations, Storage Classes.	
Unit-2:	8
Arithmetic Expressions and Precedence : Operators and Expression Using Numeric and Relational Operators, Mixed Operands, Type Conversion, Logical Operators, Bit Operations, Assignment Operator, Operator precedence and Associatively. **Conditional Branching:** Applying if and Switch Statements, Nesting if and Else and Switch.	
Unit-3:	8
Iteration and Loops: Use of While, do While and for Loops, Multiple Loop Variables, Use of Break , Goto and Continue Statements. **Arrays:** Array Notation and Representation, Manipulating Array Elements, using Multi Dimensional Arrays. Character Arrays and Strings, Structure, union, Enumerated Data types, Array of Structures, Passing Arrays to Functions.	
Unit-4:	8
Functions: Introduction, Types of Functions, Functions with Array, Passing Parameters to Functions, Call by Value, Call by Reference, Recursive Functions. **Basic of searching and Sorting Algorithms:** Searching & Sorting Algorithms (Linear Search , Binary search , Bubble Sort, Insertion and Selection Sort)	
Unit-5:	8
Pointers: Introduction, Declaration, Applications, Introduction to Dynamic Memory Allocation (Malloc, Calloc, Realloc, Free), String and String functions , Use of Pointers in Self-Referential Structures, Notion of Linked List (No Implementation) **File Handling:** File I/O Functions, Standard C Preprocessors, Defining and Calling Macros and Command-Line Arguments.	

Course Outcome:

Course Outcome (CO)		Bloom's Level
At the End of Course , the Student will be Able to Understand		
CO 1	To Develop Simple Algorithms for Arithmetic and Logical Problems.	K_2, K_3
CO 2	To Translate the Algorithms to Programs & Execution (in C Language).	K_3
CO 3	To Implement Conditional Branching, Iteration and Recursion.	K_3
CO 4	To Decompose a Problem into Functions and Synthesize a Complete Program Using Divide and Conquer Approach.	K_4
CO 5	To Use Arrays, Pointers and Structures to Develop Algorithms and Programs.	K_2, K_3

K_1- Remember, K_2- Understand, K_3- Apply, K_4- Analyze , K_5- Evaluate , K_6- Create

Text Books:

1. Schaum's Outline of Programming with C by Byron Gottfried , McGraw-Hill
2. The C programming by Kernighan Brain W. and Ritchie Dennis M., Pearson Education .
3. Computer Basics and C Programming by V.Rajaraman , PHI Learning Pvt. Limited, 2015.
4. Computer Concepts and Programming in C, E Balaguruswami, McGraw Hill
5. Computer Science- A Structured Programming Approach Using C, by Behrouz A. Forouzan, Richard F. Gilberg, Thomson, Third Edition , Cengage Learning - 2007.
6. Let Us C By Yashwant P. Kanetkar.
7. Problem Solving and Program Design in C, by Jeri R. Hanly, Elliot B. Koffman, Pearson Addison-Wesley, 2006.
8. Programming in C by Kochan Stephen G. Pearson Education – 2015.
9. Computer Concepts and Programming in C by D.S. Yadav and Rajeev Khanna, New Age International Publication.
10. Computer Concepts and Programming by Anami, Angadi and Manvi, PHI Publication
11. Computer Concepts and Programming in C by Vikas Gupta, Wiley India Publication
12. Computer Fundamentals and Programming in C. Reema Thareja, Oxford Publication

B.Tech First Year: Regular Course Lecture Plan Session 2022-23

Subject Name	Programming for Problem Solving

Unit No.	Unit Name	Syllabus Topics	Lecture No
1	Introduction to Programming: Introduction to components of a computer system:	Memory, processor, I/O Devices, storage	1
		Operating system and it's type, Introduction to low level & high level languages	2
		IDE role for development , Concept of assembler, compiler, Interpreter, loader and linker	3
	Idea of Algorithm:	Representation of Algorithm, Flowchart	4
		Pseudo code with examples	5
	Programming Basics:	Structure of C program: writing and executing the first C program, types of errors	6
		Components of C language: Standard I/O in C	7
		Fundamental data types, Variables and memory locations	8
2	Arithmetic expressions & Conditional Branching: Arithmetic expressions and precedence:	Operators and expression using numeric and relational operators, mixed operands	9
		Logical operators, bit operations, assignment operator	10
		Operator precedence and associatively, Implicit and explicit type conversion	11
	Conditional Branching:	Applying if	12
		Nesting if else	13
		Else if ladder	14
		Switch statements, use of break and default with switch.	15
3	Loops & Functions: Iteration and loops:	Use of while and for loops	16
		Concept of do while loop, break and continue	17
		Nested Loop and multiple loop variables	18
	Functions:	Introduction to function and types of functions	19
		Call by value and call by reference	20
		Storage Class	21
		Recursive functions	22

B.Tech First Year: Regular Course Lecture Plan Session 2022-23

Subject Name	Programming for Problem Solving

Unit No.	Unit Name	Syllabus Topics	Lecture
4	Arrays & Basic Algorithms: Arrays:	Array notation and representation, manipulating array elements	23
		Using multi dimensional arrays	24
		Functions with array Passing arrays to functions	25
		Character arrays and strings	26
		Structure & Union	27
		Enumerated data types	28
	Basic Algorithms:	Searching : Linear Search	29
		Binary Search	30
		Basic Sorting Algorithms : Bubble Sort	31
		Selection Sort	32
		Insertion Sort, Notion of order of complexity	33
5	Pointer& File Handling: Pointers:	Introduction, declaration, applications of pointer	34
		Introduction to dynamic memory allocation:- malloc, calloc, realloc and free	35
		Linked List: Use of pointers in self-referential structures notion of linked list (no implementation)	36
	File handling:	File I/O functions	37
		File Programs	38
		Standard C preprocessors, Types of Preprocessor Directives	39
		Command-line arguments	40

Signature	
Name of Subject Head	Ms. Radhika Jindal

Unit-3

imp.
Qu:1 What is looping control instruction? Describe the type of looping statement in 'C' with necessary syntax. (2014-15, 2018-19)

Ans ⊙ Loops in programming comes into use when we need to execute a block of statements repeatedly.

⊙ Looping can be defined as repeating the same process multiple times until a specific condition satisfies.

⊙ Type of loops: ① for loop
② while loop
③ do-while loop

① for loop :- It is used to execute a set of statements repeatedly until a particular condition is satisfie.

Syntax :-
```
for (initialization ; condition ; updation)
                ①          ②          ③
                                    inc/dec
{
    // body of the loop
④   // statements we want to execute
}
⑤ statement x;
```

order of execution:

1 → 2 (True) → 4 → 3 → 2 (True) → 4 → 3 → 2 (False) → 5 (out of the loop)

Example: Print "Hello world" 10 times.

```
int i
for( i=1 ; i<10 ; i++)
  {
   printf ("Hello world");
  }
```

Note: To run the loop we required one loop control variable like i.

2) while loop / while statement: It is used to execute a set of statements repeatedly until a particular condition is satisfied.

Syntax:-

```
(1) initialization
(2) while (condition)
    {
(3)  // statements to be execute repeatedly ;
(4)  // updation
    }
(5) statement x;
```

Order of execution:

(1) (2) (3) (4) (2) (3) (4) (2) (5)

(2) True, (2) True, (2) false → out of the loop

Note:- Since condition is checked first in both for & while loop they are known as Entry control loop.

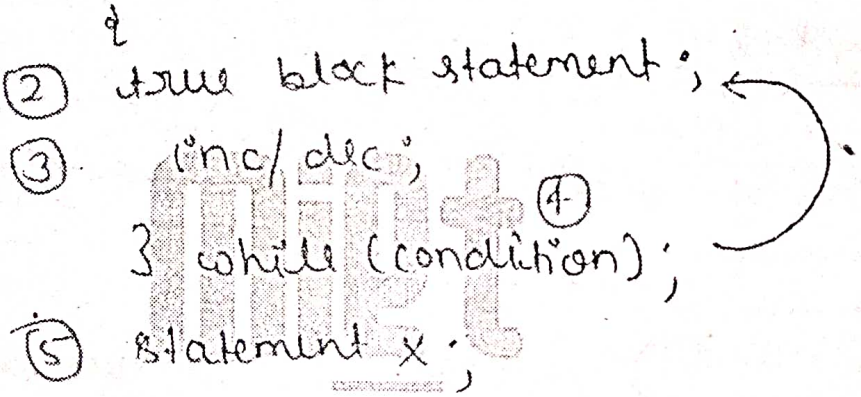
Example: Print 'a' 10 times

```
int i = 1;
while (i <= 10)
{
    printf("a");
    i = i+1;
}
```

3] Do-while loop:- Same as for & while loop.

Syntax:-

```
initialization;
do
{
    true block statement;
    inc/dec;
} while (condition);
statement x;
```



Order of execution:

1 2 3 4(True) 2 3 4(True) 2 3 4(False) 5

(out of the loop)

Example Print counting from 1 to 10.

```
int i = 1;
do
{
    printf("%d", i);
    i = i+1;
} while (i <= 10);
```

V. imp Q.2 Differentiate between do-while and while loop, (2015-16, 2018-19, 2019-20, 2020-21)

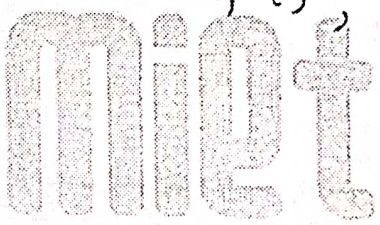
Ans

while loop	do-while loop
① It is entry control loop.	⇒ It is exit control loop.
② Here condition is checked first & if condition is true only then the block will be executed	⇒ In do-while, block is execu first & then the condition is checked so block will execu at least 1 time even condition found false.
③ In while loop, if the condition is found true for 'n' times the block will execute 'n' times.	⇒ In do-while loop, if the cond is true for n times the bloc will execute (n+1) times.
④ We donot put semicolon after while (condition) statement.	⇒ We put semicolon after whil (condition) statement.

⑤ Example
int i=1;
while (i<=n)
{
printf("%d", i);
i=i+1;
}
	Example
	int i=1
	do
	{
	printf("%d", i);
	i=i+1;
	} while (i<=n);

Ques3 Give the loop statement to print the following sequence of integer → -6, -4 -2 0 2 4 6 (2014-15)

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i;
  for( i= -6 ; i <=6 ; i++ )
     {
       if ( i%2==0)
         {
           printf( "%d ", i);
         }
     }
getch();
}
```



Ques4. Write a program to check a number is prime number or not?

Ans

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int i, num, factor=0;
   printf(" Enter number : - ");
   scanf ( "%d", &num);
   for ( i=1 ; i<=num ; i++)
```

```
    {
        if (num % i == 0)
            {
                factor++;
            }
    }
if (factor == 2)
printf("Prime number");
else
   printf("Not prime");
 getch()
}
```

Que: Write a program in 'C' to generate Fibonacci series upto the last Fibonacci number less than 100. Also find the sum of all Fibonacci numbers and total count of all Fibonacci numbers. (2014-15, 2019-20)

Sol:

```
#include<stdio.h>
#include<conio.h>
  void main()
  {
    int n, a=-1, b=1, c, i, sum=0, count=0;
     printf("Enter terms:");
     scanf("%d", &n);
     for(i=1; i<=n; i++)
        {
          c=a+b;
          if(c>=100)
```

```
        break;
    else
        {
          printf("%d", c);
          sum = sum+c;
          count = count+1;
        }
    a=b;
    b=c;
    }
printf("sum of series = %d", s);
printf("Total count of all Fibonacci number is = %d", count);
getch();
}
```

Q. Write a program in C that calculate sum of the digit of the number 2155 is 2+1+5+5 or 13. The program should accept any arbitrary number type by user (2016-17)

Sol:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num, digit, sum=0, m;
    printf("Enter any number");
    scanf("%d", &num);
    m = num;
    while(m>0)
```

```
    {
        digit = m % 10 ;
        sum = sum + digit ;
        m = m/10 ;
    }
printf( " sum of %d = %d " , num , sum) ;
getch() ;
}
```

imp.

Que7 Write a program in 'c' that will read a positive numb from the keyboard & print it in reverse order. (2015-16, 2016-17)

Ex: 24578 O/p - 87542

Sol^n

```
# include <stdio.h>
# include <conio.h>
  void main( )
   {
int num, digit , rev=0 , m;
printf ( " Enter number ");
scanf ( " %d" , & num ) ;
    m=num;
  while( m >0)
  {
    digit = m %10 ;
    rev = rev * 10 + digit ;
    m = m/10 ;
  }
printf( "Reverse of %d = %d " , num , rev);
getch() ;
}
```

Q18. Write a program to check number is palindrome or not. The program should accept any arbitary number typed by user. (2016-17, 2017-18, 2018-19, 2019-20, 2021

Sol:

```
#include <stdio.h>
#include <conio.h>
void main()
{
int num, digit, r=0, m;
printf("Enter any number");
scanf("%d", &num);
m=num;
while(m>0)
{
digit = m%10;
r = r*10 + digit;
m = m/10;
}
if(num == r)
printf("Palindrome");
else
printf("Not palindrome");
getch();
}
```

imp.
Qu9- A number is said to be an Armstrong number if the sum of the cube of its digit is equal to the number itself. For example 153 is an armstrong no. as $153 = 1^3 + 5^3 + 3^3$.

○ Write a program to check whether a given number is Armstrong or not. (2015-16, 2016-17)

Sol^n

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n, d, s=0, m;
  printf("Enter number");
  scanf("%d", &n);
   m=n;
  while(m>0)
  {
    d = m%10;
    s = s+(d*d*d);
    m = m/10;
  }
  if(m==s)
   printf("Armstrong");
  else
   printf("Not armstrong");
  getch();
}
```

Qu10- what is the concept of break statement. How to use break statement in 'c'. Explain with an example.

OR

Write a c program using while loop to elaborate the use of break statement. (2018-19) Even/odd

Sol^n

```
#include<stdio.h>
void main()
{
  int i=0;
  while(i<=100)
  {
    if(i==3)
    {
      break;
    }
    printf("%d",i);
    i=i+1;
  }
  printf("out of the loop");
  getch();
}
```

O/P → 0 1 2 out of the loop

Qu11- what is the concept of continue statement. Explain its syntax with the help of example. (2020-21)

Ans When a continue statement encountered inside a loop, control jumps to the beigning of the loop for next iteration, skip the execution of current statement for current Iteration.

Syntax:-
```
if (condition)
{
  continue;
}
```

Example:
```
int i;
for(i=1; i<=10; i++)
{
  if (i==7)
    continue;
  printf("%d", i);
}
```

Output: 1 2 3 4 5 6 8 9 10

V.imp.
Ques12 What is the difference between break and continue in C (2014-15, 2018-19, 2019-20)

Ans

Break	Continue
(1) It is used to come out of the loop.	=> It is used to come in the beginning of loop.
(2) break will skip all the coming iteration	=> while continue will skip only current iteration
(3) It is used to come out of switch statement	=> Not used in switch statement
Ex:- `int i; for(i=1; i<=3; i++) { if(i==2) break; printf("%d", i); }` O/P = 1	Ex:- `int i; for(i=1; i<=3; i++) { if(i==2) continue; printf("%d", i); }` O/P - 1 3

Q13. Discuss nesting of loop with example?

Ans When one loop is placed in block of another loop & the iteration of inner loop is based on outer loop it is termed as nesting of loop.

Ex:-
```
int i, j;
for( i=1 ; i<=3 ; i++)
  (1)     (2)    (3)
{
   for( j=1 ; j<=3 ; j++)
        (4)    (5)    (6)
   {
   (7) printf("%d", j);
   }
(8) printf("%d", i);
}
(9) Statement x;
```


Order of execution: (1) (2) (4) (5) True (7) (6) (5) True (7) False (8) (3) (2) True (4) (5) True (7) (6) --- (5) False (8) (3) (2) False (9)

Q14. Write a program to print all prime numbers from 1 to 300. (2015-16)

'OR'

Write a program to display prime numbers between 1 and 100. (2016-17)

Sol^n
```
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
  int n, i, j;
  for(n=1; n<=300; n++)
    {
      j=0;
      for(i=1; i<=n; i++)
        {
          if (n%i==0)
            j = j+1;
        }
      if(j==2)
      printf("%d", n);
    }
  getch();
}
```


Ques Write a program to find the Armstrong number from 100 to 999. (2017-18)

Ans

```
void main()
  {
    int n, d, s, m;
    for(n=100; n<=999; n++)
      {
        s=0;
        m=n;
      while(m>0)
```

```
    {
      d = m % 10 ;
      s = s + ( d * d * d) ;
      m = m / 10 ;
    }
  if ( n == s)
  printf (" % d ", n) ;
 }
 getch() ;
}
```

Q. WAP to add first seven terms of the following series by using loop $\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \cdots + \frac{7}{7!}$ (2016-17)

```
A. void main ( )
{
  int n, j , i , f , s = 0 ;
  printf (" Enter value of n") ;
  scanf (" % d ", & n) ;
  for ( j = 1 ; j <= n ; j++)
  {
    f = 1 ;
    for ( i = 1 ; i <= j ; i++)
    {
      f = f * i ;
    }
    s = s + j / f ;
  }
  printf (" sum of series = % d ", s) ;
  getch() ;
```

Ques: Write a program to print the following pattern (2020-21)

```
* * * *
* * *
* *
*
```

Sol:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n, i, j;
  printf("Enter rows");
  scanf("%d", &n);
  for(i=n; i>=1; i--)
  {
    for(j=1; j<=i; j++)
    {
      printf("* ");
    }
    printf("\n");
  }
  getch();
}
```

Write a program to print following pattern : (2016-17)

```
A
A B
A B C
A B C D
A B C D E
A B C D E F
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i, j;
  for(i=1; i<=6; i++)
  {
    for(j=65; j<=64+i; j++)
    {
      printf("%c", j);
    }
    printf("\n");
  }
  getch();
}
```

Q.11 Write a program to print following pattern :- (2017-18)

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
```

Sol^n

```
# include < stdio.h >
# include < conio.h >
Void main( )
{
  int i, j;
  for( i= 5 ; i >=2 ; i--)
  {
    for( j=1 ; j <=i ; j++)
    {
      printf( " %d ", j);
    }
  printf( " \n");
  }
  getch( );
}
```

Q2. Write a program to print with following pattern! - (2018-19)

```
10
9  8
7  6  5
4  3  2  1
```

Sol^n

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i, j, k=10;
  for(i=1; i<=4; i++)
  {
    for(j=1; j<=i; j++)
    {
      printf("%d", k--);
    }
    printf("\n");
  }
  getch();
}
```

Q.21- write a program to print the following pattern:

```
2 3 4 5 6 7
3 4 5 6 7
4 5 6 7
5 6 7
6 7
7
```

Sol:

```
#include<stdio.h>
void main()
{
    int a[10], i=0, j, count=0, k, num;
    printf("Enter a number");
    scanf("%d", &num);
    while(num>0)
    {
      count++;
      a[i] = num%10;
      num = num/10;
       i++;
    }
    for(i=count-1; i>=0; i--)
    {
    for(j=i; j>=0; j--)
      {printf("%d", a[j]);
    }
    printf("\n");
}
```

Q2. what is meant by modular programming approach? (2016-17)

Sol^n Some programs might contain thousands or millions of lines and to manage such programs it become quite difficult as there might be too many of syntax errors or logical errors present in the program, so manage such type of program, concept of modular programming approached.

It is a software design technique that emphasizes seprating the functionality of a program into independent interchangable module, such that each contains everything necessary to execute one aspect of the desired functionality

It is a process of sub dividing a computer program into seprate sub programs/module. It can often be used in a variety of applications & functions.

Characteristics:

Limitation: Limitation of each & every module should be decided i.e each module should do only 1 task.

Communication: It should be exists among different module of the program for proper execution of the entire program.

- A module can be called by one & only one higher module.
- No communication can take place directly b/w modules that do not have calling-called relationship.

(3) Way:- All module are designed as single entry and single exit system using control instructions.

imp.
Qu23- What is function? Why programmer use functions in code? Explain need of function? (2018-19), (2015-16)

Ans A function is a self contained block of code (statements) that performs a particular task.

Need of function: ○ It facilitates top-down modular programming from high level logic to lower level function.

○ The length of source program can be reduced by using functions and saves lot of time.

○ It also increase readability & it is easy to debug and find out the error.

○ It support concept of reusability & can be use any number of times.

Qu24- What are the types of function? (2014-15)

Ans (1) Library function
(2) User difined function

1) Library function: ○ There functions are already defined in the system library.

○ User must be aware of syntax of predifined function if he/she want to use the existing code in the system libraries.
Ex:- printf(), scanf(), getch() etc.

2) User defined function: ⊙ These functions are defined by the programmer or user.

⊙ Syntax of the function is given by the user so no need to include any header file.

Ex:- add(), fact(), fib() etc - -

Category of function:-

1) ⊙ Function without argument and no return values.

2) Function with argument and with return value.

3) Functions without argument and with return value.

4) Functions with argument and no return values.

V. imp.

Que25. What is the meaning of prototype of a function ?

'AND'

Explain function declaration and difinition of a function with example. (2015, 16, 17, 18, 19, 20)

Sol^n A function prototype is simply the declaration of a function that specifies function's name, parameter and return type.

⊙ The function prototype are used to tell the compiler about the number of arguments and about the ~~number~~ required datatype of a function parameter, it also tells the return type of the function.

Elements of user defined function:

1. Function declaration / Function prototype
2. Function calling
3. Function definition

1] Function declaration: function_type function_name(argum

Ex: int add (int m, int n);

↓ function prototype

The calling program should declare a function that is to be used later in the program. This is known as the function declaration or function prototype.

2] Function calling: function_name (actual argument);

Ex: add (a, b);

In order to use this function we need to invoke it at required place in the program. This is known as function calling.

3] Function definition: function_type function_name(formal argument

{

local variable declaration;

statement;

return (expression);

}

The function definition is an independent program modules that is specially written to implement the requirement of the function, it is also known as function implementation.

Q26. Difference between actual and formal arguments. (2014-15, 2016-17, 2017-18)

Ans

Actual Argument	Formal Argument
⊙ The variable which are used during function call by the calling function are known as actual parameters.	⊙ The variable which are used in function definition to collect the value of actual argument in called function are known as formal parameter.

Ex1 While executing a function, how the values are passed b/w calling & called environment? (2018-19)

Ans

```
#include <stdio.h>
int add(int x, int y); // function declaration
void main()
{
  int a, b, sum;
  printf("enter any two numbers");
  scanf("%d%d", &a, &b);
  sum = add(a,b); // function call
                  actual argument
```

```
    printf (" sum = %d ", sum);
}
int add ( int x , int y )
{
    int d;
    d = x+y;
    return (d);
}
```

(formal argument)

imp.

Q4 27. Distinguish between int main() and void main()? (2016-17)

Ans

int main()	void main()

① `#include <stdio.h>`
`int main()`
`{`
`statements;`
`return 0;`
`}`
	① `#include <stdio.h>`
	`void main()`
	`{`
	`statements;`
	`}`

② The return type of main() is int. That is the function expect a return type value int (integer) to be passed to it.	② In the above syntax, the void is the return type of the function void means null in C.
③ The return statement return value 0 to main to indicate that program has executed sucessfully.	③ The function does not return any value to the operating system after its execution.

⊙ The purpose of returning a value 0 to main is for the OS, a return of 0 means sucessful execution & anything else means there was a problem.	⊙ Since no value is return back to the OS, there is no method to know if the program ran sucessfully or not.

Que: write a function in C language to find the reverse of a given integer number? (2015-16)

Ans.

```
#include <stdio.h>
#include <conio.h>
    int rev(int);   // function prototype
     void main()
    {
     int num, t;
     printf("Enter any number");
     scanf("%d", &num);
         t = rev(num);  // function calling
printf("Reverse of %d is %d", num, t);
  getch();
}
  int rev( int n)  // function definition)
```

```
{
    int n=0, d;
    while (n > 0)
    {
        d = n % 10;
        n = n * 10 + d;
        n = n / 10;
    }
    return (n);
}
```

Imp.

Que29. Difference between call by value and call by reference with example.

Ans

Call by value	Call by reference
① In this method, values of actual parameter are passed to function's formal parameter.	→ In this method, address of actual parameter are passed to function's formal parameter.
② No use of pointer	② Use of pointer variable
③ Two types of arguments are stored in different memory locations, so any changes made inside formal parameter are not reflect in actual argument.	③ Both the actual & formal parameter refer to the same location, so any changes made inside the formal parameter are reflected in actual parameter.

WAP to swap the value of two variable using call by value

```
#include <stdio.h>
include <conio.h>
void swap( int, int );
 void main( )
  {
    int a, b;
printf(" Enter two No's");
scanf( "%d%d", &a, &b);
 swap( a, b);
 getch( );
}
void swap ( int a, int b )
{
printf( " Before swap " );
printf( " a = %d, b = %d", a,b);
    a = a+b;
    b = a-b;
    a = a-b;
printf( " After swap " );
printf( " a = %d, b = %d", a,b);
}
```

Ex: WAP to swap the value of two variable using call by reference / address ?

```
#include <stdio.h>
#include <conio.h>
  void swap( int *, int *);
   void main( )
  {
   int a, b;
printf( " Enter two numbers");
scanf( "%d %d ", &a, &b);
 swap( &a, &b );
  getch( );
}
void swap (int *n, int *y)
{
  printf ( " Before swap");
  printf ("a = %d, b = %d", *n, *y);
  *n = *n + *y;
  *y = *n - *y;
  *n = *n - *y;
printf ( " After swap " );
printf( " n = %d, y = %d", *n, *y);
}
```

(*) One advantage of the call by reference is that it is using pointer, so there is no doubling of the memory used by the variable. This is of course good, lowering the memory foot print is always a good thing. But in most cases you want the data to be private and that someone calling a function only be able to change if you want it.

So it is better to use call by value by default and only use call by reference if data changes are expected.

(Jul 30- Write a program in C that computes the area and circumference of a circle with radius taken as input using call by reference in function. (2019-20)

Sol^n

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

void AP( float *, float *, int);
void main()
{
  int r;
  float area, peri;
  printf("Enter radius of circle");
  scanf("%d", &r);
  AP( &area, &peri, r);
  printf("Area of circle = %f", area);
  printf("perimeter = %f", &peri);
  getch();
}
```

```
void AP ( float *a, float *p, int r )
{
    *a = 3.14 * r * r ;
    *p = 2 * 3.14 * r;
```

imp 3

Q What is storage class? Describe automatic, register, static and external with real syntax. (2014, 2015, 2016, 2017, 2018)

Q What do you mean by scope & lifetime of a variable? (2015-16, 2018-19)

Q Differentiate between static and register class in C language. (2016-17)

Ans Storage class in C decides the part of storage to allocate memory for a variable.

- All variable defined in 'C' get some physical location in memory where variable's value is stored.
- Memory & CPU register are the memory location where a variable's value can be stored.

⇒ Storage class of a variable in C determines

- lifetime of a variable whether it is global or local.
- Scope of the variable (visibility)
- storage location of a variable whether store in memory (RAM) or CPU register.
- Initial value of a variable.

⇒ Type of storage class

1. Automatic
2. Register
3. static
4. External

Note!- If no storage class specify then by default it is automatic.

Storage class	Storage	Initial value	Scope	Lifetime
auto	memory	Garbage	Local	till block exec
register	Register	Garbage	Local	till block execu
static	memory	zero	Local	value of variab persist blw fu call
extern	memory	zero	Global	till program execution.

⇒ Storage class example !

register storage

```
void count();
void main()
{
  clrscr();
  count();
  count();
  count();
```

static storage

```
void count();
void main()
{
  clrscr();
  count();
  count();
  count();
```

```
getch();
}
void count()
{
register int i=1;
printf("%d", i);
i = i+1;
}
```

Output 1 1 1

Here register variable are initialized every time & value of variable is deleted when function control returns.

```
getch();
}
void count()
{
static int i=1;
printf("%d", i);
i = i+1;
}
```

Output 1 2 3

Here static variables are initialized only once & value of variable is also not deleted when function control returns.

Mp

What are the main principle of recursion? Explain in detail. (2014-15, 2015-16)

Differentiate recursion and Iteration (2018-19)

The three basic principle of recursion:-

A recursive program must call itself recursively.

A recursive program must have a base condition

A recursive program must change its state & move towards the base case.

Recursion: A function is calling itself again and again to solve a particular problem is called recursion.

Base condition: In recursion, function is calling again & again so we need a termination point / condition to stop function calling, this condition is known as base condition.

If recursive program does not have base condition then the program may run for infinite time because function calling is for infinite times. This is the situation when stack overflow error occurs in recursion.

V. imp.

Q.33 Write a C program to find the factorial of a given number using recursion. (2014-15, 2018-19, 2020-21)

Ans
```
#include <stdio.h>
#include <conio.h>
int fact(int);
void main()
{
  int num, ans;
  printf("Enter a number");
  scanf("%d", &num);
  ans = fact(num);
  printf("factorial = %d", ans);
  getch();
}
```

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

V.imp.

Write a C program to generate fibonacci series using recursion.

Soln

```
# Include <stdio.h>
# include <conio.h>
  int fib( int);
  void main( )
{
  int i, num;
  printf (" Enter number of terms ");
  scanf ("%d", &num);
  printf( " Fibonacci series : = ");
  for( i=0 ; i<num ; i++)
  {
    printf( "%d ", fib(i));
  }
  getch( );
}
  int fib(int n)
{
  if(n==0 || n==1)
  return(n);
  else
  return( fib(n-1) + fib(n-2));
}
```

5 Year's

University Paper Questions

(AKTU Question Bank)

5 Years AKTU University Examination Questions		Unit-3	
S. No	Questions	Session	Lecture No
1	Describes about the types of looping statements in 'C' with necessary syntax.	2014-15(Odd)	16-22
2	A number is said to be an Armstrong number if the sum of the cube of its digit is equal to the number itself. For example, 153 is an Armstrong number as 153= 1^3+5^3+3^3.Write a C program to check whether a given number is Armstrong or not.	2015-16(Odd), 2016-17(Odd), 2016-17(Even)	16-22
3	Write a program to check number is palindrome or not. The program should accept any arbitrary number typed by user.	2017-18(Odd), 2019-20(Odd), 2020-21(Odd), 2016-17(Even)	16-22
4	Write a program to check a number is prime number or not.	2017-18(Odd)	16-22
5	Write the syntax format for while, do while and for loops.	2018-19(Odd)	16-22
6	Write a program in C to generate the Fibonacci series up to the last Fibonacci number less than 100. Also find the sum of all Fibonacci numbers and total count of all Fibonacci numbers.	2019-20(Odd), 2014-15(Even)	16-22
7	Give the loop statement to print the following sequence of integer. -6 -4 -2 0 2 4 6	2014-15(Even)	16-22
8	Write a program in 'C' that will read a positive number from the keyboard and print it in reverse order. Ex. 24578 output: 87542	2015-16(Even) 2016-17(Even)	16-22
9	Write a program that calculate sum of the digits of an integer. For example, the sum of the digit of the number 2155 is 2+1+5+5 or 13. The program should accept any arbitrary number typed by user.	2016-17(Even)	16-22
10	Differentiate between do-while and while loop.	2015-16(Odd), 2019-20(Odd), 2020-21(Odd) 2018-19(Even)	16-22
11	Why we use do-while loop in c? Also tell any properties which you know?	2017-18(Odd)	16-22
12	What is the difference between break and continue in 'C'?	2014-15(Odd), 2019-20(Odd)	16-22
13	How to use break statement in c? Explain with some sort of code.	2018-19(Odd)	16-22
14	Write the syntax of continue statement.	2020-21(Odd)	16-22
15	What is the use of break statement in loops? Write a program in 'C' using while loop to elaborate the use of break statement.	2018-19(Even)	16-22
16	Write a program to print all prime numbers from 1 to 300.	2015-16(Odd)	16-22
17	Write a C program to add first seven terms of the following series using for loop. 1/1! +2/2! +3/3! +-------	2016-17(Odd)	16-22
18	Write a program to print following pattern: A AB ABC ABCD ABCDE ABCDEF	2016-17(Odd)	16-22

19	Write a program in 'C' to display prime numbers between 1 and 100.	2016-17(Even)	16-22
20	Write a program to find the Armstrong number from 1 to 100.	2017-18(Even)	16-22
21	Write a program to print following pattern: 2 3 4 5 6 7 3 4 5 6 7 4 5 6 7 5 6 7 6 7 7	2019-20(Odd)	16-22
22	Write a program to print following pattern: * * * * * * * * * *	2020-21(Odd)	16-22
23	Write a program to print following pattern: A B C D E F G F E D C B A A B C D E F F E D C B A A B C D E E D C B A A B C D D C B A A B C C B A A B B A A A	2015-16(Even)	16-22
24	Write a program to print following numbers structure: 12345 1234 123 12	2017-18(Even)	16-22
25	Write a program to print with following pattern with appropriate comments:- 10 9 8 7 6 5 4 3 2 1	2018-19(Even)	16-22
26	What are the types of function?	2014-15(Odd)	16-22
27	What is the need of function?	2015-16(Odd)	16-22
28	What is the meaning of prototype of a function?	2015-16(Odd), 2019-20(Odd), 2017-18(Even)	16-22
29	Write a function in C language to find the reverse of a given integer number.	2015-16(Odd)	16-22
30	What are functions? What is the advantage of using multiple functions in a program?	2016-17(Odd)	16-22
31	Distinguish between – int main() and void main()?	2016-17(Odd)	16-22
32	Differentiate between actual and formal arguments.	2016-17(Odd), 2017-18(Odd), 2014-15(Even)	16-22
33	Explain function declaration and definition of a function with example.	2017-18(Odd)	16-22
34	Write a function to interchange the two values of two variables without using third variable.	2017-18(Odd), 2016-17(Even)	16-22
35	What is function? Why programmer use functions in code?	2018-19(Odd)	16-22

36	What is meant by modular programming approach?	2016-17(Even)	16-22
37	What do you mean by parameter passing? Discuss various types of parameter passing mechanism in C with example?	2016-17(Odd), 2015-16(Even)	16-22
38	Explain the difference between parameter passing mechanism call by value and call by reference. Which is more efficient and why?	2017-18(Odd), 2014-15(Even), 2016-17(Even)	16-22
39	While executing a function, how the values are passed between calling and called environment?	2018-19(Odd)	16-22
40	Write a program in C that computes the area and circumference of a circle with radius taken as input using call by reference in functions.	2019-20(Odd)	16-22
41	What is storage class? Describe automatic, register, static and external with neat syntax.	2014-15(Odd), 2017-18(Odd) 2015-16(Even)	16-22
42	Discuss the following: (i) Scope of variable (ii)Lifetime of a variable	2015-16(Odd) 2018-19(Odd)	16-22
43	Differentiate between static and register storage class in C language.	2016-17(Even)	16-22
44	Write a C program to find the factorial of a given number using recursion.	2014-15(Odd), 2018-19(Odd), 2020-2021(Odd)	16-22
45	What is recursion? Write a C program to generate the Fibonacci Series.	2016-17(Odd), 2017-18(Odd), 2016-17(Even), 2017-18(Even)	16-22
46	What are the main principles of recursion? Explain in detail.	2014-15(Even), 2015-16(Even)	16-22
47	Differentiate recursion and iteration.	2018-19(Even)	16-22