I/IV B.Tech (Supplementary) DEGREE EXAMINATION

December 2016 **Common for all branches**

First Semester **Computer programming with "C"**

Time: Three Hours **Maximum: 60 Marks**

Answer Question No 1 Compulsorily. (1 X 12 = 12 Marks)

Answer ONE Question from each unit. (4 X 12 =48 Marks)

1. Answer all questions. (12* 1= 12M)
   a) List out primitive data types.
   b) Define "constant".
   c) What syntax for Decision statement.
   d) What is the output of the following code?

```
void main()
{       int p=6;
        do      printf("Hello");
        While(p<4); }
```

   e) What is the use of an array?
   f) Give the difference between 'break' and 'continue'.
   g) What is void pointer?
   h) What is recursion?
   i) Define pointer?
   j) Differentiate Structure and Union?
   k) What is the use of getc and putc?
   l) Explain the significance of EOF.

UNIT I

2. a) Explain the structure of a C program. 6M
   b) Write a program to find the largest value among two numbers using the conditional operator 6M

(OR)

3. a) List and explain various operators used in C with examples. 6M
   b) Explain about various MATH functions with C programs with examples. 6M

UNIT II

4. a) What is dynamic array? How is it created? Give example. 6M
   b) Write a program using do-while loop to calculate and print 'm' Fibonacci numbers 6M

(OR)

5. a) Explain the declaration of single dimension and two dimensional arrays. 6M
   b) Write a program to perform matrix multiplication. 6M

UNIT III

6. a) What is user defined functions? Discuss with example. 6M
   b) Write a recursive function to find GCD value. 6M

(OR)

7. a) What are the storage classes supported by C language? Explain each in detail 6M
   b) Write a C program to implement binary search. 6M

UNIT IV

8. a) What is structure? Explain structures within structures with suitable examples. 6M
   b) Explain the following with examples
      i) Unions ii) Bit fields iii) The size of operator. 6M

(OR)

9. a) Explain the concept of file I/O. 6M
   b) Explain the command line arguments with an example. 6M

I/IV B.Tech (Supplementary) DEGREE EXAMINATION

DEC 2016	**Common to all branches**
Third Semester	**Computer programming with "C"**
Time: Three Hours	**Maximum: 60 Marks**
Answer Question No 1 Compulsorily.	(1 X 12 = 12 Marks)
Answer ONE Question from each unit.	(4 X 12 =48 Marks)

Scheme of Evaluation & Solutions

1 a) List out primitive data types.

Ans: The primitive data types are

Char, int, float and void

b) Define "constant".

Ans: const is a keyword. It is used to create constant values in a c program OR

A Constant is a fixed value, which doesn't change during the execution of the program.

c) Write syntax for decision statement

Ans:

The syntax for decision statement is

```
if( condition)
        True statements (or) body of if
else
        False statements (or) body of else
```

[Award 1 mark for any decision making statement]

d) What is the output of the following code?

Ans:

The output of the given code is

Hello

e) What is the use of an array?

Ans: The use of an array is to hold or store group of similar data values (or) information.

[Any relevant answer -1M]

f) Give the difference between 'break' and 'continue'.

Ans:

break statement simply skips the execution of remaining statements(present in same control statement) and transfers the controller to immediate statement followed by loop. Continue statement suspend the execution of the loop for that iteration and transfer control back to the loop for the next iteration.

g) What is void pointer?

Ans:

void pointer is a generic pointer. We can refer all types of information by type casting.

h) What is recursion?

Ans: Recursion is the process of calling same function itself.

i) Define pointer.

Ans: pointer is a variable that holds the address of another variable of same type.

j) Differentiate structure and union

Ans:

The size of structure variable is greater than or equal to the sum of sizes of its members. The size of union variable is equal to the size of largest member.

k) What is the use of getc and putc?

Ans:

getc() is used to read a character from a file.

Putc() is used to store a character in a file.

l) Explain the significance of EOF.

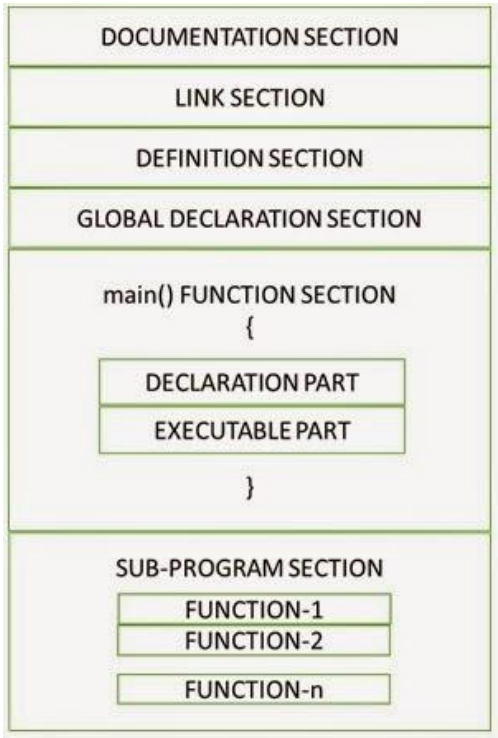
Ans: The significance of EOF is to determine the end of the file.

UNIT - I

2.a) Explain the structure of a c program. 6M

[Structure – 2M, Explaining any 4 sections- 2M, relevant example program -2M]

Ans: Structure of C Program



Every C program contains a number of building blocks known as functions. Each function performs a specific task independently. A C program comprises following different sections.

1. **Include header file section:**
   - C program depends upon some header files for function defitions that are used in the program.
   - Each header file has the extension '.h'.

   Example:

   #include<ctype.h>

2. **Global declaration section:**
   - This section declares some variables that are used in more than on function
   - This section must be declared outside of all the functions.
3. **main() function:**
   - Every program in C must contain main() function.
   - The execution of a c program starts at the beginning of this function.
   - The prototype of a main() function is
     int main(void)
4. **Declaration part:**
   - This part contains the declaration of all the local variables that are used in executable part.
   - The initialization of variables can also be done in this section.
5. **Executable part:**
   - This part contains the statements following the declaration of variables.
6. **User defined function :**
   - The functions defined by the user are called user defined functions.
   - These functions are defined outside the main() function.
7. **Comments:**
   - Comments are not necessary in a program. However to understand the program documentation is needed..

2. b) Write a program to find the largest value among two numbers using the conditional operator.6M
[Preprocessor directives -1M, Variable declaration- 1M, I/O statements -2M, Logic 2M]

Ans:

```
#include<stdio.h>
int main()
{
        inta,b,c;
        printf("\n Enter two numbers");
        scanf("%d%d",&a,&b);
        c=(a>b)?a:b;
        printf("\nlargest number is %d",c);
        return 0;
}
                        (OR)
#include<stdio.h>
int main()
{
        inta,b;
        printf("\n Enter two numbers");
        scanf("%d%d",&a,&b);
         (a>b)?printf("\nlargest is %d",a):printf("\nlargest is %d",b);
        return 0;
}
```

3.a) List and explain various operators used in C with examples. 6M

[Operator Def- 1M, Listing operators- 1M, Explaining any 4 operators -4M]

Ans:

An operator is a symbol that tells the compiler to perform specific operations. For the better understanding operators in C are classified into 7 types

i. Arithmetic operators
ii. Relational operators
iii. Logical operators
iv. Increment and decrement operators
v. Bitwise operators
vi. Conditional operator
vii. Assignment operators

The description of all operators is followed

Arithmetic operators:

Arithmetic operators are used to perform arithmetic operations.

Operator	meaning	example
+	addition	A+B
-	Substraction	A-B
*	Multiplication	A*B
/	division	A/B
%	modulo division	A%B

Relational operators:

These operators are used to test the conditions i.e. compare the values both sides to the operator and returns either 0 or 1 based on the desired result.

'C' programing language has **six** relational operators, those are

Operator	meaning	example
<	less than	A<B
>	greater than	A>B
>=	greater than or equal	A>=B
<=	less than or equal to	A<=B
==	equal to	A==B
!=	not equal to	A!=B

Logical operators

Logical operators are used to test more than one condition in same statement. 'C' has three logical operators those are

Operator	meaning	example
&&	AND	A>B && A!=B
||	OR	A==B || A<B
!	NOT	!(A<B)

Increment and decrement operators:

These operators are used to increase or decrease the value of operand by 1. The symbols ++is used for performing increment operation and **--**was used for performing decrement operation.

Operator	meaning	example
++	increment	A++
- -	decrement	A--

Bitwise operators

These operators are used to perform operations upon bits i.e. to do bit manipulations.

Operator	meaning	example
&	Bitwise AND	A&B
|	Bitwise OR	A|B
<<	Left shift	A<<B
>>	Right shift	A>>B
^	Bitwise XOR	A^B
~	One's compliment	~(A)

Conditional operator

The conditional operator is used to test the relation between variables. The conditional operator contains condition followed by two statements or values. If the condition is true the first statement is executed otherwise second statement is executed.

The syntax of the conditional operator is

Condition? Expression1 : expression2;

The other name for conditional operator is ternary operator.

Assignment operators

The assignment operator is used to assign a value to the variable.

Operators

= += -+ *= <<= >>= /= &= != %=

3.b) Explain about various MATH functions with C programs with examples. 6M

[Purpose of math functions- 1M, Listing functions- 1M, Explaining any 4 functions -4M]

Ans:

The C language provides set of mathematical functions to do mathematical operations. The following are the mathematical functions available in C language.

pow(): It calculates power of a value.
- Syntax: pow(value1,value2).
- Example: pow(x,y) – finds x to the power y.

sqrt(): It calculates positive square root of the given value.
- Syntax: sqrt(value).
- Example: sqrt(x) – finds square root of x.

Log(): calculates natural logarithm of given value with base 10.
- Syntax: log(value).
- Example: log(x) – finds log value of x.

Exp(): it calculates exponential of a value.
- Syntax: exp(value).
- Example: exp(x) – finds exponential function to x.

Floor(): It finds largest integer not greater than the given value.
- Syntax: floor(value).
- Example: floor(2.345) =2.

Ceil(): it finds the smallest integer not greater than given value.

Syntax: ceil(value).
Example: ceil(4.67) = 5.

Abs(): it finds absolute value of its argument.
Syntax: abs(value).
Example: abs(-2) = 2.

NOTE: The above mathematical functions are defined in math.h header file.

4.a) What is dynamic array? How is created? Give example. 6M

[Def- 1M, syntax- 1M, Explaination -2M, Any relevant example program-2M(preprocessor directives -1M, dynamic memory allocation-1M)]

Ans:

The size of the array change at runtime is called as dynamic array. We create dynamic arrays in C program using malloc() and calloc() dynamic memory allocation functions.

Dynamic array Declaration:

Daya_type *array_name;
n<-number of elements to be read
array_name=(data_type*)malloc(n*sizeof(datatype));

Example:

```
#include<stdio.h>
void main(){
        int *a,i,n;
        printf("\nenter size");
        scanf("%d",&n);
        a=(int*)malloc(n*sizeof(int));
        printf("\nenter elements");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        printf("\nelements are");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
}
```

Note: Any relevant program –Marks will be awarded.

4.b) Write a program using do-while loop to calculate and print 'm' Fibonacci numbers. 6M

[Preprocessor directives -1M, main & variable declaration 1M, I/O statements -2M, Logic 2M]

Ans:

```
#include<stdio.h>
void main(){
int f1=0,f2=1,f,l,i=0;
printf("\nenter length of the series");
scanf("%d",&l);
printf("%d\t%d",f1,f2);
do{
        f=f1+f2;
        printf("\t%d",f);
        f1=f2;f2=f;
        i++;
```

```
        }while(i<l-2);
}
```

Note: Any relevant program –marks can be awarded.

5.a Explain the declaration of single dimension and two dimensional arrays. 6M

[Single dimensional array-3M(syntax -1M, examples-1M,Explanation-1M), Two dimensional array – 3M(syntax -1M, examples-1M,Explanation-1M)]

Ans:

An array is a collection of elements of similar type, those sharing a common name and occupies contiguous locations.

Declaration of a one dimensional array

The syntax of the declaration of a one dimensional array is

Data_type arrayname[size];

Here data_type refers to the type of the elements in the array and it can be a basic data type. Size is an integer expression representing total number of elements in the array.

In general arrayname[i] refers to the i th element of the array.

Example:

```
#include<stdio.h>
Void main( )
{
        int a[10], i, n ;
        printf (" Enter how many values you want to read : ");
        scanf (" %d ", &n );
        for( i=0; i<n; i++ )
        {
                printf ( " Enter the value of a[%d] : " , i );
                scanf (" %d ", &a[ i ]);
        }
        printf (" The array elements are ; ");
        for( i=0; i<n; i++ )
                printf (" \t %d ", a[ i ]);
}
```

Declaration of a two dimensional array

The syntax of the declaration of a two dimensional array is

Data_type arrayname[size1][size2];

Here data_type refers to the data type of the elements in the array and it can be a basic data type. Size1 is an integer expression representing the row size in the array and Size2 is also an integer expression representing the column size in the array. In a two dimensional array two subscripts are used in two pairs of square brackets. The total number of elements in a two dimensional array is calculated by multiplying number of rows and the number of columns. In a two dimensional array the values are stored in a row major order.

Example:

```
#include<stdio.h>
Void main( )
{
        int a[10][10], i, j, m, n ;
        printf (" Enter the number of rows and columns: ");
```

```
        scanf (" %d %d", &m, &n );
        for( i=0; i<m; i++ )
        {
                for( j=0; j<n; j++ )
                {
                        printf ( " Enter the value of a[%d][%d] : " , i, j );
                        scanf (" %d ", &a[ i ][ j ]);
                }
        }
        printf (" The array elements are ; ");
        for( i=0; i<m; i++ )
                for( j=0; j<n; j++)
                        printf (" \t %d ", a[ i ][ j ]);
}
```

NOTE: Any suitable example programs---marks can be awarded.

5.b Write a program to perform matrix multiplication. 6M

[Preprocessor directives -1M, main & variable declaration 1M, I/O operations -2M, Logic 2M]

Ans:

```
#include<stdio.h>
void main()
{
        int a[20][20],b[20][20],c[20][20],i,j,k;
        intm,n,p,q;
        printf("\nenter order of first matrix");
        scanf("%d%d",&m,&n);
        printf("\nenter order of second matrix");
        scanf("%d%d",&p,&q);
        if(n==p)
        {
                printf("\nenter elements into first matrix");
                for(i=0;i<m;i++){
                for(j=0;j<n;j++){
                        scanf("%d",&a[i][j]);
                 }
                }
                printf("\nenter elements into second matrix");
                for(i=0;i<p;i++){
                for(j=0;j<q;j++){
                        scanf("%d",&b[i][j]);
                }
                }
                for(i=0;i<m;i++){
                for(j=0;j<n;j++){
                        c[i][j]=0;
                        for(k=0;k<q;k++){
```

```
                                        c[i][j]=c[i][j]+a[i][k]*b[k][j];
                                }
                        }
                        }
                        printf("\n multiplication of two matrices is\n");
                        for(i=0;i<m;i++){
                        for(j=0;j<n;j++){
                                printf("%d\t",c[i][j]);
                        }
                printf("\n");
                        }
                }
                else
                printf("\nmultiplication is not possible");
        }
```

6.a) What is user defined function? Discuss with an example. 6M

[Definition -1M, Examples-1M, Types of user defined functions-4M(each one carries 1M)]

The functions which are declared and defined by user are called as user defined functions.

1. Function with arguments and with return value:

The functions under this category have some arguments and it return some value to the calling function.

Syntax:

```
Return type  <function name>(arguments)
{
        Statements;
}
```

Example:

```
#include<stdio.h>
int add(int,int);
void main()
{
        int a=2,b=3,c;
        c=add(a,b);
        printf("\nsum is %d",c);
}


void add(intx,int y)
{
        returnx+y;

}
```

2. Function with arguments and without return value:

The functions under this category have some arguments and it does not return any value to the calling function.

Syntax:

```
void<function name>(arguments)
{
        Statements;
}
```

Example:

```
#include<stdio.h>
void add(int,int);
void main()
{
        int a=2,b=3;
        add(a,b);
}
void add(intx,int y)
{
        printf("\nsum is %d",x+y);
}
```

3. Function without arguments and with return value:

The functions under this category don't have any arguments and it return some value to the calling function.

Syntax:

```
Return type <function name>()
{
        Statements;
}
```

Example:

```
#include<stdio.h>
int add();
void main()
{
        int c=add();
        printf("\nsum is %d",c);
}
int add()
{
        int a=2,b=5;
        returna+b;
}
```

4. Function without arguments and without return value:

The functions under this category don't have any arguments and it does not return any value to the calling function.

Syntax:

```
void<function name>()
{
        Statements;
```

```
}
```

Example:

```
#include<stdio.h>
void add();
void main()
{
        add();
        printf("\n bye");
}
void add()
{
        int a=2,b=5;
        printf("\nsum is %d",a+b);;
}
```

NOTE: Marks can be allotted for any relevant examples.

6.b) Write a recursive function to find GCD value. 6M

[Recursion def-1M, Program -5M (Preprocessor directives -1M , main & variable declaration 1M, I/O statements -2M, recursive function- 2M)]

Ans: Recursion is a repetitive process in which a function calls itself.

Program:

```
#include<stdio.h>
int gcd(int,int);
void main(){
        int a,b,c;
        printf("\nenter two numbers");
        scanf("%d%d",&a,&b);
        c=gcd(a,b);
        printf("\ngcd of %d and %d is %d",a,b,c);
}
int gcd(int x,int y){
        if(y==0)
                return x;
        else
                return gcd(y,x%y);
}
```

7.a) What are the various storage classes supported by C language? Explain each in detail. 6M

[definition-1M, listing storage classes-1M,explain each class-4M]

Ans:

A variable is considered to a name given to the memory location in which a constant value can be stored. To define a variable, it is not sufficient to specify the data type of the variable, but also we must specify storage class for the variable. The data type specifies which type of value it can store. Now storageclass specifies at which location the variable value is to be stored.

The storage class specifies the following:

1. Where the variable could be **saved**..
2. The **default/initial value** the variable is taking.

3. The **scope** of the variable: scope indicates in which block or in which function the variable can be formed.

4. **The life time of the variable:** lifetime indicates how long the variable exists in that function (or) in that block.

Types of storage classes:

1. automatic storage class
2. register storage class
3. static storage class
4. extern storage class

Automatic storage class:

auto keyword is used to specify automatic storage class. When a variable is created under this storage class that will become as local to that function or block.

Such variables are called as local variables.

Local variables have the following.

Memory: stack segment
Default: garbage value
Scope: block scope
Lifetime: alive until control with in that block.
Example:

```
#include<stdio.h>
void main(){
        autoint i=2;
        printf("%d",i);
}
```

Register storage class:

Variables which are declared under this class are accessed faster than variables which are declared using other storage classes.

We use register keyword to create variables under this storage class.

Memory: registers
Default: garbage value
Scope: block scope
Lifetime: alive until control with in that block.
Example:

```
#include<stdio.h>
void main(){
        register int i=2;
        printf("%d",i);
}
```

Extern storage class:

Extern keyword is used to declare variables under external storage class.

When variables are declared using extern, they becomes global variables that means their value can be accessed during entire program.

Memory: data segment
Default: zero
Scope: program scope

Lifetime: alive until control with in that program

Example:

```
#include<stdio.h>
extern int x;
void one();
voidmain(){
        printf("%d\n",x);
        x=100;
        one();
}
void one(){
        printf("%d",x);
}
```

Static storage class:

We use static keyword to create static variables under static storage class.

We use static variables to persist values of variables in between different function calls.

Memory: data segment

Default: zero

Scope: block/program scope

Lifetime: alive until control with in that program

Example:

```
#include<stdio.h>
void one();
void main(){
        int i;
        for(i=0;i<3;i++)
                one();
}
void one(){
        static int a=2;
        a++;
        printf("%d\t",a);
}
```

7.b) Write a c program to implement binary search. 6M

[Preprocessor directives -1M, main & variable declaration 1M, I/O statements -2M, Logic 2M]

Ans:

```
#include<stdio.h>
intbsearch(int[],int,int);
void sort(int[],int);
void main(){
        inti,n,e,a[20],s;
        printf("\nenter number of elements");
        scanf("%d",&n);
        printf("\nenter elements");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
```

```
printf("\nenter element to serach");
scanf("%d",&e);
sort(a,n);
s=bsearch(a,n,e);
if(s==1)
        printf("\nelement found in the list");
else
        printf("\nelement not found in the list");
}
void sort(int a[],int n)
{
        inti,j,t;
        for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
                if(a[i]>a[j]){
                t=a[i];
                a[i]=a[j];
                a[j]=t;
                }
        }
        }
}
intbsearch(int a[],intn,int e)
{
        inti,l=0,h=n-1,m;
        while(l<=h)
        {
                 m=(l+h)/2;
                if(a[m]==e)
                {
                return 1;
                break;
                }
                else if(e<a[m])
                h=m-1;
                else
                 l=m+1;
        }
}
```

8.a) What is structure? Explain structures within structures with suitable examples. 6M

[Definition-1M, structure with in the structure- 1M, example -1M, Relevant example program -3M]

Ans:

The structure is user defined data type that allows user to store collection of heterogeneous(different) types of data under some common name.

structure within structure:

structure with in another structure is called nested structure. The purpose of this concept is to use information of one structure in another structure.

Syntax:

```
struct tagname1{
        member 1;
        member n;
}
struct tagname2{
        member 1;
        member n;
        struct tagname1 variable;
}
```

Example:

```
#include<stdio.h>
void main(){
        struct date{
                intdd,mm,yy;
        };
        struct employee{
                intssn;
                charename[20];
                struct date doj;
        }e;
        printf("\nenter employee name and number");
        scanf("\n%s%d",e.ename,&e.ssn);
        printf("\nenter date of joined");
        scanf("%d%d%d",&e.doj.dd,&e.doj.mm,&e.doj.yy);
        printf("\nemployee details are\n");
        printf("name\t%s\nssn\t%d\ndoj\t%d/%d/%d",
                e.ename,e.ssn,e.doj.dd,e.doj.mm,e.doj.yy);
}
```

8.b) Explain the following with examples. **6M**

i) Unions ii) Bit fields iii) The size of operator.

[**unions- 2M, Bit fields- 2M, Sizeof -2M**](expected: definition, syntax, and usage.)

Ans:

Unions:

union is used to create user defined data types; union is a collection of different types of data stores under some common name. Programmer declares unions in his program by using a key word called "union".

Declaration syntax:

```
union union-name{
        Datatype member-name 1;
        Datatype member-name 2;
                ……
        Datatype member-name n;
```

```
};
```

- The size of union variable is equal to the size of largest member
- Each member in a union is assigned a shared memory area location
- Only one member can be accessed at a time.

Bit fields:

Bit fields are used to assign less number of bits to the unsigned or int members of structure or union.

Syntax:

```
Struct tagname{
        Unsigned member:N;
};
```

N refers number of bits to be assigned to that member.

Example:

```
Void main()
{
        struct ta{
                int no:2;
        }b;
        b.no=400;
        printf("%d",b.no);
}
```

Size of operator:

this operator returns the size of the variable that occupies in the memory in terms of bytes.

Syntax:

```
Sizeof(variable);
```

Example:

```
#include<stdio.h>
void main()
{
        char a;
        printf("\nsize of character is %d",sizeof(a));
}
```

9.a) Explain the concept of file I/O. 6M

[File def-1M, listing file handling functions with purpose-3M, explain any 2 functions-2M]

Ans:

File is a collection of information that can store input or output. Using files we can store large volumes of data at a time permanently.

- Reading and writing data of file will be done using file input/output functions.
- C provides the following I/O functions.

Output functions:

fgetc(), fgets(), fgetw(), fscanf().

Input functions:

fputc(), fputs(), fputw(), fprintf().

fgetc():- This reads a character from a file.

Syntax: char fgetc(file pointer);

fgets():- this reads a string from a file.

Syntax: fgets (char*,file pointer);

fgetw():- this reads an integer from a file.

Syntax: intfgetw(file pointer);

fscanf():- this reads formatted data from a file.

Syntax: fscanf(filepointer,"format string",arguments);

Fputc(): This writes a character into a file.

Syntax: fputc(char,filepointer);

Fputw(): This writes an integer into a file.

Syntax: fputw(int,filepointer);

Fputs(): This writes a string into a file

Syntax: fputs(char*,filepointer);

Fprintf(): This writes formatted data into a file.

Syntax: fprintf(filepointer,"format string",arguments);

9.b) Explain command line arguments with an example. 6M

[Command line arguments def-1M, example-1M, relevant program -4M(Structure-1M, logic-1M, Concept of command line arguments- 2M)]

Ans:

- The arguments those are passed to the main function of your program from the operating system command line at the time of execution are known as command line arguments or command line parameters.
- Main function takes two arguments. First argument was **argc**(argument count) of integer type it indicates number of arguments is passed. Second argument is **argv**(argument value) of character array of pointer type.
- The main () routine can check argc to see how many arguments the user specified.
- The program can find out its own name as it was invoked: it is stored in the argv[0] string.

Example:

```
#include<stdio.h>
#include<stdlib.h>
void main(int argc, char *argv[])
{
        int n1,n2;
        n1= atoi(argv[1]);
        n2=atoi[argv[2]);
        printf("number of arguments is: %d", argc);
        printf("some is :%d", n1+n2);
}
```

******

HOD,IT.

Scheme Prepared by.

Signature of Paper evaluators:

S.No	Name of the Faculty	Name of the college	Signature
1			
2			
3			
4			
5			
6			
7			