C – PROGRAMMING

MODEL PAPER QUESTIONS

SAQ'S

1. **a) *Define Operating System.***

Ans. Operating system: It is the interface between user and hardware. These programs are in-built into the computer and are used to govern the control of a computer hardware components such as processors, memory devices and Input and Output devices.

Ex:- Linux, Android, IOS, Windows, etc..

b) List keywords in C language.

Ans. Keywords in C : Every keyword in C is classified as either a 'keyword' or an 'identifier'.

• All keywords have fixed meanings and these meanings cannot be changed. All keywords must be written In lowercase.There are 32 keywords in C.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	Short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C) Show the use of jump statement.

Ans. Jump Statement : In C, jump statements are used to jump from one part of the code to another altering the normal flow of the program. They are used to transfer the program control to somewhere else in the program.

Types of Jump Statements in C :

There are 4 types of jump statements in C:

1. Break
2. Continue
3. Goto
4. Return

1) <u>**Break in C**</u>

The break statement exits or terminates the loop or switch statement based on a certain condition, without executing the remaining code.

<u>Syntax of break in C :</u>

```
break;
```

2) <u>**Continue in C**</u>

The continue statement in C is used to skip the remaining code after the continue statement within a loop and jump to the next iteration of the loop. When the continue statement is encountered, the loop control immediately jumps to the next iteration, by skipping the lines of code written after it within the loop body.

<u>Syntax of continue in C :</u>

```
continue;
```

3) <u>**Goto Statement in C**</u>

The goto statement is used to jump to a specific point from anywhere in a function. It is used to transfer the program control to a labeled statement within the same function.

<u>Syntax of goto Statement :</u>

```
goto label;

.

.

Label:

//code
```

4) **<u>Return Statement in C</u>**

The return statement in C is used to terminate the execution of a function and return a value to the caller. It is commonly used to provide a result back to the calling code.

<u>Syntax for return statement :</u>

return expression;

d) *Outline a few library functions.*

<u>Ans.</u> <u>Library Functions in C :</u> The standard library functions are built-in functions in C programming to handle tasks such as mathematical computations, I/O processing, string handling etc.

• These functions are defined in the header file. When you include the header file, these functions are available for use.

For example:

• The **printf**() is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in “stdio.h” header file.

• There are other numerous library functions defined under “stdio.h”, such as scanf(), fprintf(), getchar() etc. Once you include “stdio.h” in your program, all these functions are available for use.

e) *What is macro preprocessor ?*

<u>Ans.</u> <u>Macros in C</u>: Macros are pieces of code in a program that is given some name. Whenever this name is encountered by the compiler, the compiler replaces the name with the actual piece of code. The ‘#define’ directive is used to define a macro.

<u>Syntax of Macro preprocessor</u> :

#define token value

Where after preprocessing, the token will be expanded to its value in the program.

f) *Define malloc() function.*

Ans. **Malloc**() **Function:** The “malloc” or “memory allocation” method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.

Syntax :

Ptr = (cast-type*) malloc(byte-size);

Example:

Ptr = (int*) malloc(100 * sizeof(int));

•Since the size of int is 4 bytes, this statement will allocate 400 bytes of memory. And, the pointer ptr holds the address of the first byte in the allocated memory.

LAQ'S

2 a) ***Explain different operators in C.***

Ans. **Operators in C :** C supports a rich set of operators. Operators are used in programs to manipulate data and variables. They usually form a part of the mathematical of logical expressions.

C operators are classified into a number of categories. They include:

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and Decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

1)**ARITHMETIC OPERATORS**

The Arithmetic operators are

+ (Addition)

-(Subtraction)

* (Multiplication)

/ (Division)

% (Modulo division)

Eg: 1) a+b 2) a-b 3) a*b 4) a/b 5) a%b

Note: C does not have any operator for exponentiation.

2) **RELATIONAL OPERATORS**

Comparisons can be done with the help of relational operators. The expression containing a relational operator is termed as a relational expression.The value of a Relational Expression is either zero or 1.

Operator	meaning
1) <	(is less than)
2) <=	(is less than or equal to)
3) >	(is greater than)
4) >=	(is greater than or equal to)
5) = =	(is equal to)
6) !=	(is not equal to)

Eg: a<b or 1<20

3) **LOGICAL OPERATERS**

C has the following three logical operators.

Operator	Meaning
&&	(logical AND)
||	(logical OR)
!	(logical NOT)

Eg: 1) if(age>55 && sal<1000)

2)If(number<0 || number>100)

4) **ASSIGNMENT OPERATORS**

The usual assignment operator is '='.In addition, C has a set of 'shorthand' assignment operators of the form,

v op = exp;

Eg: x += y+1;

This is same as the statement

X=x+(y+1);

5) **INCREMENT AND DECREMENT OPERATORS**

C has two very useful operators that are not generally found in other languages. These are the Increment and decrement operators:

++ and –

The operator ++ adds 1 to the operands while – subtracts 1.It takes the following form:

++m; or m++ --m; or m—

6) **CONDITIONAL OPERATOR**

A ternary operator pair "?:" is available in C to construct conditional expression of the form:

SYNTAX:

exp1 ? exp2 : exp3;

Here exp1 is evaluated first. If it is true then the expression exp2 is evaluated and becomes the value of the expression. If exp1 is false then exp3 is evaluated and its value becomes the value of the expression.

7) **BITWISE OPERATORS**

Bitwise operator are used for manipulation of data at bit level. These operators are used for testing the bits or shifting them right or left bitwise operators may not be applied to float or double.

Operator	Meaning

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift right
~	One’s complement

8) **SPECIAL OPERATORS**

C supports some special operators such as

comma operator

Size of operator

Pointer operators(& and *) and

Member selection operators(. and ->)

2. b) Construct an algorithm for sum of individuals (digits).

Ans. Algorithm for sum of digits :

```
1. Start
2. Initialize sum = 0
3. Input a number
4. Use while loop
5. Repeat until number is not zero

   while (N1= 0)

6. Remainder = N % 10

   Sum = sum + Remainder

   N = N/10

7. Print sum
8. Stop.
```

3. *a) Explain Arithmetic expression and Precedence.*

Ans. ARITHMETIC EXPRESSIONS

An arithmetic expression is a combination of variables, constants, and operators arranged as per the syntax of the language.

Ex: a=x+y;

PRECEDENCE OF OPERATORS :

Precedence, in C, is the rule that specifies the order in which certain operations need to be performed in an expression. For a given expression containing more than two operators, it determines which operations should be calculated first.

In C, precedence of arithmetic operators(*, %, /, +, -) is higher than relational operators(==, !=, >, <, >=, <=) and precedence of relational operator is higher than logical

operators(&&, || and !).

Example of precedence :

(1 > 2 + 3 && 4)

This expression is equivalent to:

((1 > (2 + 3)) && 4):

i.e, (2 + 3) executes first resulting into 5 then,

first part of the expression (1 > 5) executes resulting into 0 (false)

then, (0 && 4) executes resulting into 0 (false).

Output :

0

3 b) Illustrate a program of calculator using switch. Statement.

Ans. Program for Implementation of arithmetic calculator using Switch Case Statement :

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

```
#include <conio.h>
 void main ()
 {
 int a, b, sub, mul, add, div, mod, opt;
printf (“Menu:\n 1.Addition\n 2. Subtraction \n 3.Multiplication \n 4.Division \n
5.Modulo \n”);
printf (“Enter option from menu: \n”);
scanf (“%d”, &opt);
 printf (“\n Enter two nos:”):
scanf (“%d %d”. &a, &b);
switch (opt)
{
case 1: add= a + b;
 printf (“\n sum is=%d”, add);
break;
case 2 : sub= a -b;
 printf (“\n difference is=%d”, sub);
break:
 case 3: mul= a* b:
 printf (“\n Product is=%d", mul):
 break;
 case 4: div= a /b;
 printf (“\n Quotient is=%d”, div):
 break:
 case 5: mod= a % b;
 printf (“\n Remainder is=%d”, mod):
```

```
 break;
 default: printf ("\n Wrong Option!!!");
 break;
}
 getch();
}
```

```
Output:
Menu:
 1.Addition
 2. Subtraction
3.Multiplication
4.Division
5.Modulo
 Enter option from menu:
3
Enter two nos: 12
5
Product is=60.
```

---

4 a) Build a difference between Actual and Formal parameters.

<u>Ans</u>.

<u>Actual Parameters</u>	**<u>Formal Parameters</u>**
1.When a function is called, the values(expressions) that are passed in the function call	1. The parameter used in Function definition Statement which contain

are called the arguments or actual parameters.	On its time of declarat- ion are called formal Parameters.
2. These are the variables or expressions referenced in the parameter list of a sub-program call.	2. These are the variables or expressions referen- ed in the parameter list ication.
3. Actual Parameters are the parameters which are in calling sub program.	**3.** Formal Parameters are the parameters which are in called sub program
4. There is no need to specify the datatype in actual parameter.	**4.** The datatype of the receiving value must Be defined.
5. The parameters are written in function call and are known as actual. Parameters.	**5.** The parameters are written in function def -inition are known as Formal Parameters.
6. Actual Parameters can't be Constant values or variable Names.	**6.** Formal Parameters can be treated as local variab -les of a function in which They are used in the Function header.

---

4 b) Construct a program of Fibonacci's Series.

Ans. **Fibonacci's Series :** The Fibonacci series is the sequence where each number is the sum of the previous two numbers of the sequence. The first two numbers are 0 and 1 which are used to generate the whole series.

// C Program to print the Fibonacci series using recursion

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

```
// Recursive function to print the fibonacci series
Void fib(int n, int prev1, int prev2) {
 // Base Case: when n gets less than 3
 If (n < 3) {
 Return;
 }

 Int curr = prev1 + prev2;
 Prev2 = prev1;
 Prev1 = curr;
 Printf("%d ", curr);
 Return fib(n – 1, prev1, prev2);
}
// Function that handles the first two terms and calls the
// recursive function
Void printFib(int n) {
 // When the number of terms is less than 1
 If (n < 1) {
 Printf("Invalid number of terms\n");
}
 // When the number of terms is 1
 Else if (n == 1) {
 Printf("%d ", 0);
 }
 // When the number of terms is 2
 Else if (n == 2) {
```

```
 Printf(“%d %d”, 0, 1);
 }
 // When number of terms greater than 2
 Else {
 Printf(“%d %d “, 0, 1);
 Fib(n, 0, 1);
 }
 Return;
}
Int main() {
 Int n = 9;
 // Printing first 9 fibonacci series terms
 printFib(n);
 return 0;
}
```

---

5 a) Define Union. Demonstrate with Student example.

Ans. **Union** : The Union is a user-defined data type in C language that can contain elements of the different data types just like structure. But unlike structures, all the members in the C union are stored in the same memory location. Due to this, only one member can store data at the given instance.

Example :

```
//program for Student Information
#include <stdio.h>
union student {
 char firstName[50];
 int roll;
```

```
float sub1marks,sub2marks,sub3marks,avg;
} s[];
Int main() {
 Int I;
 float totalmarks,avg,grade;
 printf("Enter information of students:\n");
 // storing information
 for (I = 0; I < 5; ++i) {
 S[i].roll = I + 1;
 printf("\nFor roll number%d,\n", s[i].roll);
 printf("Enter first name: ");
 scanf("%s", s[i].firstName);
 printf("Enter subject1 marks: ");
 scanf("%f", &s[i].sub1marks);
 printf("Enter subject2 marks: ");
 scanf("%f", &s[i].sub2marks);
 printf("Enter subject3 marks: ");
 scanf("%f", &s[i].sub3marks);
 }
 printf("Displaying Information:\n\n");
 // displaying information
 for (I = 0; I < 5; ++i) {
 printf("\nRoll number: %d\n", I + 1);
 printf("First name: ");
 puts(s[i].firstName);
printf("subject1 Marks: %f\n", s[i].sub1marks);
```

```
 printf(“subject2 Marks: %f\n”, s[i].sub2marks);

 printf(“subject3 Marks: %f\n”, s[i].sub3marks);

 printf(“\n”);

 S[i].avg=s[i].sub1marks+s[i].sub2marks+s[i].sub3marks/3;

 Printf(“average marks of student:%f\n”,s[i].avg);

 }

 return 0;

}
```

5 b) Construct and display an employee Union with Ename,

Eid, Eage, Esal, DOJ.

__**Ans.**__ // C Program to Construct and display an employee Union with Ename, Eid, Eage, Esal, DOJ.

```
#include <stdio.h>

#include <string.h>

// Declaration of the

// dependent structure

Struct Employee

{

 Int employee_id;

 Char name[20];

 Int salary;

};

// Declaration of the

// Outer structure

Struct Organisation
```

```
{
 Char organisation_name[20];
 Char org_number[20];
 // Dependent structure is used
 // as a member inside the main
 // structure for implementing
 // nested structure
 Struct Employee emp;
};
// Driver code
Int main()
{
// Structure variable
 Struct Organisation org;
 // Print the size of organisation
 // structure
 Printf("The size of structure organisation : %ld\n",
 Sizeof(org));
 Org.emp.employee_id = 101;
 Strcpy(org.emp.name, "Robert");
 Org.emp.salary = 400000;
 Strcpy(org.organisation_name,
 "GeeksforGeeks");
 Strcpy(org.org_number, "GFG123768");

 // Printing the details
 Printf("Organisation Name : %s\n",
 Org.organisation_name);
 Printf("Organisation Number : %s\n",
 Org.org_number);
 Printf("Employee id : %d\n",
 Org.emp.employee_id);
 Printf("Employee name : %s\n",
 Org.emp.name);
 Printf("Employee Salary : %d\n",
 Org.emp.salary);
}
```

Output:

```
The size of structure organisation : 68
Organisation Name : GeeksforGeeks
Organisation Number : GFG12376
Employee id : 101
Employee name : Robert
Employee Salary : 400000
```

6 a) Name different types of pointers.

Ans. 1. NULL Pointer :

The Null Pointers are those pointers that do not point to any memory location. They can be created by assigning a NULL value to the pointer. A pointer of any type can be assigned the NULLvalue.

Syntax:

```
Data_type *pointer_name = NULL;
```

(or)

Pointer_name = NULL;

• It is said to be good practice to assign NULL to the pointers currently not in use.

2. **<u>Generic Pointer :</u>**
   A void pointer is a pointer that has no associated data type with it. A void pointer can hold an address of any type and can be typecasted to any type.
   • A void pointer is a pointer that has no associated data type with it. A void pointer can hold an address of any type and can be typecasted to any type.

   • Void pointers cannot be dereferenced.

   • The C standard doesn't allow pointer arithmetic with void pointers. However, in GNUC it is allowed by considering the size of the void as 1.

3. **<u>Double Pointers :</u>**
   In C language, we can define a pointer that stores the memory address of another pointer. Such pointers are called double-pointers or pointers-to-pointer. Instead of pointing to a data value, they point to another pointer.
   <u>Syntax:</u>
   Datatype ** pointer_name;

6 b) Select few random access file functions.

<u>Ans.</u> <u>Functions of Random Access file :</u>

Consequently, there are mainly three functions that assist in accessing the random

<u>Access file in C</u>:

- ftell()
- rewind()
- fseek()

1. ***<u>ftell() function</u>***:

The file pointer's position is relative to the file's beginning and can be determined using the ftell() function.

Syntax:

It has the following syntax:

```
ftell(FILE *fp)
```

2. ***rewind() function***:

The file pointer can be moved to the file's beginning using the rewind() function. When a file needs to be updated, it is useful.

Syntax:

It has the following syntax:

```
rewind(FILE *fp);
```

3. *fseek() function*:

The fseek() function is used to move the file position to a given location.

Syntax:

The syntax is:

int fseek(FILE *fp, long displacement, int origin);

Constant	Value	Position
SEEK_SET	0	Beginning of file
SEEK_CURRENT	1	Current position
SEEK_END	2	End of file

7 a) Explain in detail about FILE Handling Functions.

Ans. C File Operations :

C file operations refer to the different possible operations that we can perform on a file in C such as:

1. Creating a new file – fopen() with attributes as "a" or "a+" or "w" or "w+"
2. Opening an existing file – fopen()

3. Reading from file – fscanf() or fgets()
4. Writing to a file – fprintf() or fputs()
5. Moving to a specific location in a file – fseek(), rewind()
6. Closing a file – fclose()

• <u>Create a File in C :</u>

The fopen() function can not only open a file but also can create a file if it does not exist already. For that, we have to use the modes that allow the creation of a file if not found such as w, w+, wb, wb+, a, a+, ab, and ab+.

<u>Syntax:</u>

FILE *fptr;

Fptr = fopen("filename.txt", "w");

• <u>Reading From a File</u>

The file read operation in C can be performed using functions fscanf() or fgets(). Both the functions performed the same operations as that of scanf and gets but with an additional Parameter, the file pointer. There are also other functions we can use to read from a file. Such Functions are listed below:

Function	*Description*
fscanf()	Use formatted string and variable arguments list to take input from a file.
fgets()	Input the whole line from the file.
fgetc()	Reads a single character from the file.
fgetw()	Reads a number from a file.
fread()	Reads the specified bytes of data from a binary file.

• <u>Write to a File :</u>

The file write operations can be performed by the functions fprintf() and fputs() with similarities to read operations. C programming also provides some other functions that can be Used to write data to a file such as:

Function	*Description*
fprintf()	Similar to printf(), this function use formatted string and variable argument list to print output to the file.
fputs()	Prints the whole line in the file and a newline at the end.
fputc()	Prints a single character into the file.
fputw()	Prints a number to the file.
fwrite()	This functions write the specified amount of bytes to the file.

• **Closing a File :**

The fclose() function is used to close the file. After successful file operations, you must always close a file to remove it from the memory.

Syntax of fclose() :

fclose(file_pointer);

• where the file_pointer is the pointer to the opened file.

7 b) List different data types in detail in C.

Ans. Data types in C :

C language is rich in its data types. C data types are defined as the data storage format that a variable can store a data to perform a specific operation.there are three types of data types.

1. Primary (or fundamental) data types
2. User_defined data types

3. Derived data types

1. **PRIMARY DATATYPES :**
   All C compilers support fourfundamental data types.

• Integer (int)– integer: a whole number.

• Character (char) – a single character

• Floating point (float) – floating point value: ie a number with a fractional part.

•Double (double) – a double-precision floating point value.

• void – valueless special purpose type.

INTEGER DATA TYPE:

• Integer data type allows a variable to store numeric values.

• "int" keyword is used to refer integer data type.

• The storage size of int data type is 2 byte.

• It varies depend upon the processor in the CPU that we use. If we are using 16 bit processor, 2 byte (16 bit) of memory will be allocated for int data type.

• int (2 byte) can store values from -32,768 to +32,767

CHARACTER DATA TYPE:

• Character data type allows a variable to store only one character.

• Storage size of character data type is 1. We can store only one character using character data type.

• "char" keyword is used to refer character data type.

• For example, 'A' can be stored using char datatype. You can't store more than one character using Char data type.

Note : To store more than one characters in a variable we have to use string .

FLOAT DATA TYPE:

• Float data type allows a variable to store real values.

• Storage size of float data type is 4. This also varies depend upon the processor in the CPU as “int” data type.

• We can use up-to 6 digits after decimal using float data type.

• For example, 10.456789 can be stored in a variable using float data type.

DOUBLE:

• Double data type is also same as float data type which allows up-to 14 digits after decimal.

Datatype	*Range of values*
char	-128 to 127
int	-32, 768 to 32, 767
float	3.4e-38 to 3.4e+e38
double	1.7e-308 to 1.7e+308

2. **User defined datatypes** :
   They are classified into two types:

• Type definition(typedef)

• Enumeration datatype(enum)

• **Type Definition(typedef)**

C supports a feature known as “type definition” that allows user to define an identifier that would represent an existing data type. It takes the general form:

```
typedef type identifier;
```

type refers to an existing data type

identifier refers to name giving to data type

<u>Enumeration data type</u> is defined as follows:

enum identifier {value1,value2,....value n};

• Identifier is a user defined enumerated datatype which can be used to declare variables that can have one of the values enclosed within the braces known as enumeration constants.

Eg: enum days{mon,tue,wed,thr,fri,sat,sun};

• The compiler automatically assigns integer digits beginning with 0 to all the enumeration constants.

3. **<u>Derived data type :</u>**

The datatypes that are derived from the existing primary datatypes is called as derived datatype.

Eg: Arrays

Functions

Structures

Unions

Pointers