F.Y. Diploma : Sem. II
[DE/EJ/ET/EN/EX]

'C' Programming Language

Time : 3 Hrs.] Prelim Question Paper Solution **[Marks : 70**

Q.1 Attempt any FIVE of the following : [10]

Q.1 (a) List any four relational operators. [2]

Ans.: [Any four relational operators ½ mark each]

Four relational operators:

< Less than
> Greater than
<= Less than equal to
>= Greater than equal to
== Equal to
!= Not equal to

Q.1 (b) State uses of * and & operators with respect to pointer. [2]

Ans.: [Use of each 1 mark]

***** - It is used to declare a pointer variable.

```
int *ptr;
```

It is also used as value at operator.

```
printf("%d",*ptr);
```

&-It is used to retrieve address from the memory.

```
int a,*ptr;
ptr=&a;
```

Q.1 (c) Write output of following Program: [2]

```
# include <stdio.h>
    void main()
    { int i;
        for(i =–1; i<=10;i++)
        {
            if(i<5)
            continue;
            else
            break;
            printf("Hello\n\n");
        }
    }
```

Ans.: [Correct output 2 marks]

```
#include<stdio.h>
void main()
{
int i;
for(i=-1;i<=10;i++)
{
if(i<5)
continue;
else
break;
printf("Hello \n\n");
 }
}
```

Output : No output (Nothing will be displayed on screen as break statement takes the control out of the loop).

Q.1 (d) Define array. How one dimensional array is declared? **[2]**

Ans.: [Definition 1 mark and Declaration 1 mark]

Array: An array is a collection of data elements of same data type. The values in an array are stored in continuous memory locations.

To declare a one dimensional array:
datatype arr_name[size];
int arr[5] ;

Q.1 (e) State any two types of function on the basis of parameter passing and return type. **[2]**

Ans.: [Any two types 1 mark each]

Types of functions:

(1) Function with no return value and with parameter
(2) Function with return value and with no parameter
(3) Function with return value and with parameter
(4) Function with no return value and with no parameter

Q.1 (f) Define the term : (i) identifier (ii) token **[2]**

Ans.: (i) identifier [1 mark]

Identifier is a user-defined name and consists of a sequence of letters and digits. It refers to the names of variables, functions and arrays.
e.g. main, amount

(ii) token [1 mark]

In a program, the smallest individual unit is known as Token.
e.g. keyword, constants

Q.1 (g) State any two differences between while and do–while statement. **[2]**

Ans.: [Two points 2 marks]

while statement	do while statement
while loop checks the condition first and then executes statements.	do while loop first executes statements and then checks the condition.
If expression is true then only statements inside block are executed otherwise loop terminates.	If expression is false then also at least once statements inside block are executed.

Syntax :
while (expression)
{
Statements
}
	Syntax :
	do
	{
	Statements
	} while (expression);

Q.2 Attempt any THREE of the following: **[12]**

Q.2(a) Explain ? : operator with suitable example. **[4]**

Ans.: Conditional Operator (Ternary Operator) [2 marks]

It takes the form "?:" to construct conditional expressions
The operator "?:" works as follows :
Syntax :
exp1? exp2 : exp3;
Where exp1, exp2 and exp3 are expressions.
exp1 is evaluated first, if it is true, then expression exp2 is evaluated.
If exp1 is false, exp3 is evaluated.

Example [2 mark]

```
int a = 10, b = 5, x;
x = (a > b) ? a : b;
```

here x will take value 10 because condition given is if a > b.

Q.2(b) Explain any four bit wise operators used in C with example. **[4]**

Ans.: [Any four bitwise operator with example 1 mark each]

Bitwise operators

Bitwise OR – |

It takes 2 bit patterns, and performs OR operation on each pair of corresponding bits. The following example will explain it.

```
          1010
          1100
---------------
OR        1110
```

Bitwise AND – &

It takes 2 bit patterns, and perform AND operations with it.

```
---------
AND  1000
---------
```

The Bitwise AND will take pair of bits from each position, and if only both the bit is 1, the result on that position will be 1. Bitwise AND is used to Turn-off bits.

Bitwise NOT

One's complement operator (Bitwise NOT) is used to convert each "1–bit to 0-bit" and "0–bit to 1-bit", in the given binary pattern. It is a unary operator i.e. it takes only one operand.

```
1001
NOT  0110
----------
```

Bitwise XOR ^

Bitwise XOR ^, takes 2 bit patterns and perform XOR operation with it.

```
          0101
          0110
----------
XOR       0011
----------
```

Left shift Operator – <<

The left shift operator will shift the bits towards left for the given number of times.

```
int a = 2<<1;
```

Right shift Operator – >>

The right shift operator will shift the bits towards right for the given number of times.

```
int a = 8>>1;
```

Q.2(c) Write a program to calculate factorial of number. **[4]**

Ans.: [Correct logic 2 marks and correct syntax 2 marks]

```
#include<stdio.h>
#include<conio.h>
void main() {
int fact = 1,n,i;
clrscr();
```

```
printf("Enter a number");
scanf("%d",&n);
for(i = 1; i <= n; i++) {
fact = fact*i;
}
printf("%d",fact);
getch();
}
```

Q.2(d) Write a program to print following pattern: **[4]**

1 2 3 4
5 6 7
8 9
10

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

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

Q.3 Attempt any THREE of the following: **[12]**

Q.3(a) Write a program to sort element of an array descending order. **[4]**

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

```
#include<stdio.h>
#include<conio.h>
void main()
{
int arr[5];
int i,j,temp;
clrscr();
printf("\n Enter elements:");
for(i=0;i<5;i++)
{
scanf("%d",&arr[i]);
}
for(i=0;i<4;i++)
{
for(j=i+1;j<5;j++)
{
if(arr[i]<arr[j])
{
temp=arr[i];
```

```
arr[i]=arr[j];
arr[j]=temp;
}
}
}
printf("\nSorted array elements :\n");
for(i=0;i<5;i++)
printf("%d ",arr[i]) ;
getch();
}
```

Q.3(b) Write a program to take input as a number and reverse it by using while loop. [4]

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

```
# include<stdio.h>
# include<conio.h>
void main( )
{
int no, rem, rev = 1;
clrscr( );
printf("\n Enter number");
scanf("%d",&no);
while(no>=1)
{
rem=no%10;
printf("%d",rem);
no=no/10;
}
getch( );
}
```

Q.3(c) Write a program to print fibonacci series. [4]

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

```
# include<stdio.h>
# include<conio.h>
void main( )
{
int a,b,c,i,no;
clrscr( );
printf("\n Enter number of elements");
scanf("%d",%no);
a=0;
b=1;
printf("%d\t%d\t",a,b);
for(i=0;i<no-2;i++)
{
c=a+b;
printf("%d\t",c);
a=b;
b=c;
}
getch( );
}
```

Q.3(d) Write a program to find length of given string without library function. **[4]**

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

```
#include<stdio.h>
#include<conio.h>
void main()
{
int len,i=0;
char str1[10];
clrscr();
printf("\n Enter string :");
gets(str1);
while(str1[i]!="\0")
{
i++;
}
printf("%d",i);
getch();
}
```

Q.4 Attempt any THREE of the following: **[12]**

Q.4(a) Write difference between call by value and call by reference. **[4]**

Ans.: [Any four points 1 mark each]

Sr. No.	call by value	call by reference
(1)	In call by value, a copy of actual argument is passed to formal arguments of the called function	In call by reference the location (address) of actual argument is passed to formal arguments of called function
(2)	Any changes made to the formal arguments in called function have no effect on the values of actual argument	Any changes made to the formal argument in called function affects the values of actual argument
(3)	Pointers are not used	Pointers are used

(4)	***Example*:**
	#include<stdio.h>
	void swapbyvalue(int,int);
	int main()
	{
	int n1=10, n2=20;
	swapbyvalue(n1,n2);
	printf("n1=%d,n2=%d",n1,n2);
	}
	void swapbyvalue(int a, int b)
	{
	int t;
	t=a; a=b; b=t;
	}
		***Example*:**
		#include<stdio.h>
		void swapbyreference(int *, int *);
		int main()
		{
		int n1=10, n2=20;
		swapbyreference(&n1,&n2);
		printf("n1=%d,n2=%d",n1,n2);
		}
		void swapbyreference(int * a, int* b)
		{
		int t;
		t=*a; a=*b; *b=t;
		}

Q.4(b) Write a program for addition of two 3 × 3 matrix. **[4]**

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

```
# include<stdio.h>
# include<conio.h>
void main( )
{
int a[3][3], b[3][3],c[3][3],i,j;
```

```
clrscr( );
printf("Enter first matrix elements:\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("\nEnter second matrix elements:\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&b[i][j]);
}
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
c[i][j]=a[i][j] + b[i][j];
}
}
}
printf("\n\nAddition of two matrices is:");
for(i=0;i<3;i++)
{
printf("\n\n");
for(j=0;j<3;j++)
{
printf("%d\t",c[i][j]);
}
}
getch( );
}
```

Q.4(c) Explain recursion with example. **[4]**

Ans.: [Description of recursion 2 marks and any suitable example 2 marks]

Recursion means a function calls itself. A recursive function contains a function call to itself inside its body. A recursive function is called from main ( ) function for the first time. After that every time function is called from its body.

Example:

```
 void main ( )
{
display ( ); // first call to display function
}
void display( )
{                          //Recursive call to display function
display ( );
}
```

Q.4(d) Distinguish between global and local variable. [4]
(Note: Any other relevant point shall be considered.)

Ans.: [Any four points 1 mark each]

Sr. No.	Local Variable	Global Variable
(1)	Local variables are declared inside a function.	Global Variables are declared outside all function.
(2)	Local Variables cannot be accessed outside the function.	Global Variables can be accessed in any function.
(3)	Local Variables are alive only within a function.	Global Variables are alive till the end of the program.
(4)	Created when the function block is created and destroyed when it is deleted.	Remain in existence for the entire time a program is executing

Q.4(e) Write a program to declare a structure stationary having data member, name quantity and cost. Accept and display this information for five items. [4]

Ans.: [Correct logic 2 marks and Correct syntax 2 marks]

```
#include<stdio.h>
#include<conio.h>
struct stationery
{
char name[20];
int quantity,cost;
} s[5];
void main( )
{
int i;
clrsccr( );
printf("\n Enter information");
for(i=0;i<5;i++)
scanf("%s%d%d",&s[i].name,&s[i].quantity,&s[i].cost);
printf("\n Display information");
for(i=0;i<5;i++)
printf("%s%d%d",s[i].name,s[i].quantity,s[i].cost);
getch( );
}
```

Q.5 Attempt any TWO of the following: [12]

Q.5(a) Write a program in 'c' to copy one string into other without using built in function. [6]

Ans.: [Correct logic 3 marks and Correct syntax 3 marks]

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
char str[20];
char dest[20];
clrscr();
printf("Enter a string");
scanf("%s",&str);
for(i=0;str[i]!='\0';i++)
{
```

```
dest[i]=str[i];
}
dest[i]='\0';
printf("The source string is %s",str);
printf("\nThe copied string is %s",dest);
getch();
}
```

Q.5(b) What is array of structure? List any two differences between array and array of structure. [6]

Ans.: [Explanation of array of structure 2 marks and any 2 relevant differences 2 mark each]

Array of structure is collection of structure. Structure is used to store the information of one particular object but if want to store many objects then array of structure is used.

Example

```
struct student
{
    int roll_no;
    char name[10];
}
```

Following are differences:

- Array stores similar data type elements whereas array of structure stores variables of structure where each structure variable contains different data type elements.
- Example of array : int a[10];
  Example of Array of structure :

```
struct book
{
char name[10];
flat price;
};
struct book b[100];
```

Q.5(c) State four storage class. Explain any one. [6]

Ans.: [List of storage classes 3 marks, Explanation of any one 3 marks]

There are four storage classes in C :

(i) Automatic
(ii) Static
(iii) External
(iv) Register

Features	Automatic Storage Class	Register Storage Class	Static Storage Class	External Storage Class
Keyword	auto	register	static	extern
Initial Value	Garbage	Garbage	Zero	Zero
Storage	Memory	CPU register	Memory	Memory
Scope	scope limited, local to block	scope limited, local to block	scope limited, local to block	Global
Lifetime	limited life of block, where defined	limited life of block, where defined	value of variable persist between different function calls	Global, till the program execution
location	Memory	Register memory	memory	memory

Example	void main( )
	{
	auto int i;
	printf("%d",i);
	}
	OUTPUT
	124
		void main( )
		{
		register int i;
		for(i=1;i<=5;
		i++);
		printf("%d",i);
		}
		OUTPUT
		12345
			void add( );
			void main( )
			{
			add( );
			add( );
			}
			void add( )
			{
			static int i=1;
			printf("\n%d",i);
			i=i+1;
			}
			OUTPUT
			1
			2
				void main( )
				{
				extern int i;
				printf("%d",i);
				int i=5
				}
				OUTPUT
				5

Q.6 Attempt any TWO of the following: [12]

Q.6(a) Write a program to show use of array of pointers. [6]

Ans.: [Correct logic 3 marks and Correct syntax 3 marks]

```
# include<stdio.h>
void main()
{
int *a[4];
int i=31,j=5,k=19,l=71,m;
arr[0]=&i;
arr[1]=&j;
arr[1]=&k;
arr[1]=&l;
for (m=0; i<=3; m++)
printf("%d", *(arr[m]);
}
```

Q.6(b) State any two advantages and disadvantages of pointer. [6]

Ans.: Advantages : [Any 3 correct advantages 3 marks]

(i) Pointers are more efficient in handling arrays and data tables.
(ii) They can be used to return multiple values from a function via function arguments.
(iii) Pointers permit reference to functions and thereby facilitating passing of functions as arguments to other functions.
(iv) The use of pointer arrays to character strings results in saving of data storage space in memory.
(v) Pointers allow C to support dynamic memory management.
(vi) Pointers reduce length and complexity of programs.
(vii) They increase the execution speed and thus reduce the program execution time.

Disadvantages [Any 3 correct disadvantages 3 marks]

(i) If it contains an incorrect value it can lead to a problem when used.
(ii) When you use this incorrect pointer to read a memory location, you may be reading a incorrect garbage value then error may be occurred in the program.
(iii) Pointers are slower than normal variables.
(iv) If pointers are updated with incorrect values, it might lead to memory corruption.

Q.6(c) State four arithmetic operations perform on pointer with example. [6]

Ans.: [List of any four arithmetic operations on pointer 1 mark each, Example 2Marks]

```
int * i;
i++;
```

In the above case, pointer will be of 2 bytes. And when we increment it, it will increment by 2 bytes because int is also of 2 bytes.

```
float * i;
i– –;
```

In this case, size of pointer is still 2 bytes. But now, when we decrement it, it will decrement by 4 bytes because float is of 4 bytes.

```
int *a,*b,*c;
*a = 10;
*b=20;
*c=*a**b;
printf("%d",*c);
```

Here, Normal multiplication operation is done on pointer variables.

```
int *a,*b,*c;
*a = 10;
*b=20;
*c=*a + *b;
printf("%d",*c);
```

Here, Normal addition operation is performed on pointer variables.

❑ ❑ ❑ ❑ ❑