F.Y. Diploma : Sem. II
[CO/CD/CM/CW/IF]

Programming in 'C'

Time : 3 Hrs.] Prelim Question Paper Solution **[Marks : 100**

Q.1 Attempt any TEN of the following : **[20]**

Q.1(a) Enlist any four keyword used in C. **[2]**

(A)

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

Q.1 (b) What is operator precedence and associativity? **[2]**

(A) **Operator precedence** means the sequence in which operations are done. It means operators of higher precedence are evaluated first.

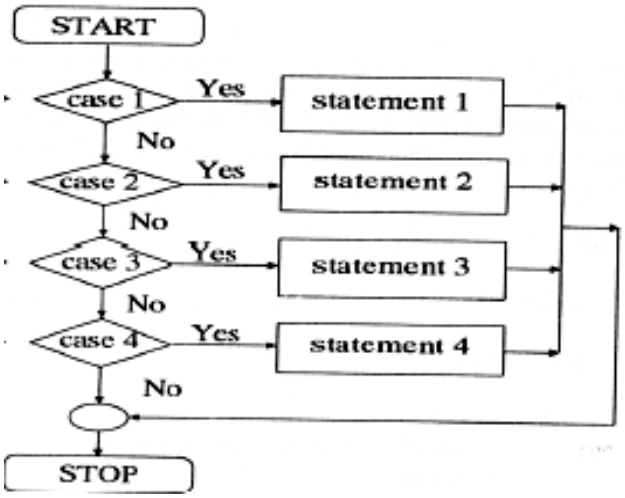
Associativity gives their evaluation order. When an expression with more than one operators having equal priority, is executed, it works according to associativity.

OR

Category	Operator	Associativity
Postfix	( ) [ ] –> . ++ – –	Left to right
Unary	+ – ! – == – – (type)* & size of	Right to left
Multiplicative	*/%	Left to right
Additive	+ –	Left to right

Q.1 (c) Draw the flow–chart of switch statement. **[2]**

(A)



Q.1 (d) Write syntax and use of **[2]**

(i) strcmp() **(ii) strcpy()**

(A) (i) strcmp()

This function compares two strings identified by arguments and returns zero if both strings are equal, otherwise it returns the difference between ASCII values of first non matching character pair from the strings.

Syntax : strcmp (string1, string2);

(ii) strcpy()

Function strcpy( ) copies the content of one string to the other sting. It takes two argumetns.

Syntax : strcpy(destination string, source string);

Q.1 (e) Find the output of the following : **[2]**

```
void main(){
    int i=1;
    clrscr();
    for(;;){
        printf("%d ",i);
    }
    getch();
}
```

(A)

```
void main(){
   int i=1;
   clrscr();
   for(;;){
      printf("%d ",i);
   }
   getch();
}
```

Ans :

Output – infinite

Q.1 (f) What is a storage class? Enlist storage classes available in C. **[2]**

(A) A storage class defines the scope and life time of variables and/or functions within a C program. There are following storage classes which can be used in a C program.

- auto
- register
- static
- extern

Q.1 (g) State the advantage of function. **[2]**

(A) The program may become too large and complex and as a result the task of debugging, testing and maintaining a program becomes difficult. So if program is divided into different functions, then each function may be independently coded and later can be combined into a single unit. Functions are easy to understand, debug and test. Hence the task becomes easy.

Q.1 (h) What is the use of (–>) operator with pointers? **[2]**

(A) To access members of structure with pointer variable, we use arrow → in between pointer variable and structure member.

Syntax : ptr → member of structure

Q.1 (i) State four rules for choosing variable name. **[2]**

(A) Rules for choosing variable name:

(i) It must begin with a letter. Some systems permit underscore as the first character.

(ii) ANSI standard recognizes a length of 31 characters. However length of variable should not normally more than eight characters, since only the first characters are treated as significant by many compilers.

(iii) Uppercase and lowercase are significant. That is, the variable Total is not the same as total or TOTAL.

(iv) Variable should not be a keyword.

(v) In variable declaration white space is not allowed.

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

(A) (i) identifier

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

In a program, the smallest individual unit is known as Token.

e.g. keyword, constants

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

(A)

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.1 (l) Find out the error if any, in the program. **[2]**

```
#include<stdio.h>
int main(){
    int i=1;
    switch(i){
        case 1: printf("\radioactive cats have 18 half-lives");
        break;
        case 1*2+4: printf("\bottle for rent -inquire within");
        break;
    }
    return 0;
}
```

(A)

```
#include<stdio.h>
int main(){
   int i=1;
   switch(i){
      case 1: printf("\radioactive cats have 18 half-lives");
      break;
      case 1*2+4: printf("\bottle for rent -inquire within");
      break;
   }
   return 0;
}
```

Ans :

No error

Q.2 Attempt any Four of the following: [16]

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

(A) Conditional Operator (Ternary Operator)

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

```
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]

(A) 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) Enlist any four types of arithmetic operators used in C and give one example of each. [4]

(A)

Operator	Example (considering a = 10, b = 2)
+ (plus)	Addition of two numbers. Eg.: c = a + b; Ans : c = 12
– (minus)	Subtractions of two numbers. Eg.: c = a – b; Ans c = 8
* (product)	Multiplication of two numbers. Eg.: c = a*b; Ans = 20
/ (division)	Dividing one number by other. Eg.: c = a/b; Ans = 5
% (modulus)	Dividing one number by other and get remainder of the division. Eg.: c = a%b; Ans = 0

Q.2(d)Distinguish between call-by-value and call-by-reference. [4]

(A)

Call by Value	Call by Reference
This is the usual method to call a function in which only the value of the variable is passed as an argument.	In this method, the address of the variable is passed as an argument.
Any alternation in the value of the argument passed is local to the function and is not accepted in the calling program.	Any alternation in the value of the argument passed is accepted in the calling program.
Memory location occupied by formal and actual arguments is different.	Memory location occupied by formal and actual arguments is same and there is a saving of memory location.
Since a new location is created, this method is slow.	Since the existing memory location is used through its address, this method is fast.
There is no possibility of wrong data manipulation since the arguments are directly used in an application.	There is a possibility of wrong data manipulation since the addresses are used in an expression. A good skill of programming is required here.

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

1 2 3 4
5 6 7
8 9
10

(A)

```
# 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.2(f) Explain with the example the use of Break statement. **[4]**

(A) The break statement in C programming language has the following two usage :

(i) When the break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

(ii) It can be used to terminate the case in the switch statement.
If one is using nested loops, the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax –

```
break;
```

Example :

```
#include <stdio.h>

int main()
{
    int a = 10;
    while(a<20)
    {
        printf("value of a: %d \n", a)
        a++;
        if(a > 15)
        {
            break;
        }
    }
        return 0;
}
```

Q.3 Attempt any Four of the following: **[16]**

Q.3(a) Write a program to display whether entered number is positive integer or not. **[4]**

(A)

```
# include<stdio.h>
# include<conio.h>
voidmain( )
{
int no;
clrscr( );
printf("\n Enter a number");
scanf("%d",&no);
if(no>0)
printf("Number is positive");
else
printf("Number is not positive");
getch( );
}
```

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

(A)

```
# 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) Explain with example array of pointer. **[4]**

(A) A pointer is a variable that contains an address which is a location of another variable in memory.

Syntax to create an array of pointers:

data type * arr_name[size];

Example:

```
int *x[2];
char *name[3]={"Nashik","Mumbai","Pune"};
```

Program:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
char *str[5]={"keyboard", "monitor", "CPU","harddisk", "SMPS"};
int i =0;
clrscr( );
for(i=0;i<=4;i++)
printf("\n %d element = %s",i,str[i]);
getch();
}
```

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

(A)

```
# 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(e) What is the difference between pre-increment and post-increment operators? Explain with the help of an example. **[4]**

(A) As the name says Pre increment means **increment first than operate**. Post increment means **operate than increment**.

eg
for x=7
a=++x
x will increase first than a=x will operate

while for
a=x++
a will be equal x(7) and after than x value will be 8.

```
#include<stdio.h>
main()
{
    int x = 7;
    int a;
    a = ++x;
    printf("the value of a = %d", a);
}
```

Output :
The value of a = 8

```
#include<stdio.h>
main()
{
    int x = 7;
    int a;
    a = x++;
    printf("the value of a = %d", a);
}
```

The value of a = 7

Q.3(f) Write a program to display all even numbers from 1 to N. **[4]**

(A)

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int N,i;
clrscr( );
printf("\n Enter value for N");
scanf("%d",&N);
for(i=1;i<=N;i++)
{
if(i%2= =0)
printf("%d\t",i);
}
getch( );
}
```

Q.4 Attempt any Four of the following: [16]

Q.4(a) Explain the declaration and initialize of character array. [4]

(A) In C character array is used to represent string. String is a group of characters.

Declaration of character array

Syntax :

char character_array_name[size];

char is a data type, character_array_name represents string name, size determines the number of automatically stores a NULL character ('\0) at the end of the string.

Example :

char name[10];

Initialization of character array :

Compile time initialization :

Character array can be initialized when it is declared.

Character array can be initialized with any of the following methods :

- char city[7]="Mumbai";
  city character array is 7 elements long as Mumbai contains 6 characters and 1 space for NULL character.
- char city[10]="Mumbai";
  city character array is declared as 10 characters long where Mumbai occupy 6 characters and remaining space is initialized with '\0' characters.
- char city[7]={'M','u','m','b','a','I','\0'};
  city character array is 7 element long as Mumbai contains 6 characters and 1 space for NULL character.
- char city[ ]={'M','u','m','b','a','I','\0'};
  character array can be initialized without specifying size. In this case, the size of the array will be determined automatically, based on the number of elements initialized. In the example size of character array city is 7.

Run time initialization

An array can be initialized using scanf( ) function at run time.

```
int a[5];
for (i=0;i<5;i++)
scanf("%d",&a[i]);
```

Q.4(b) How the size of an array is calculated? Explain with suitable example. [4]

(A) Size of an array is calculated with respect to two parameters :

(i) Memory (number of bytes) required by the data type of the variable.

(ii) Number of elements declared as size in square bracket along with variable name.

Size of an array=memory required by data type*number of elements inside array variable.

Example

```
int a[10];
```

Memory (number of bytes) required by the data type int–2 bytes.

Number of elements in array variable a – 10

Size of a=2*10=20 bytes

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

(A)

```
# 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(d) Write a syntax of switch case statement and write it's example. **[4]**

(A) Syntax:

```
switch (expression)
{
case value-1:
……..
……..
break;
case value-2:
……..
……..
break;
default:
……..
……..
```

```
break;
}
```
Here value1, value 2 are probable conditions or cases

Program:

```
#include<stdio.h>
void main()
{
int no=0;
clrscr();
printf("Enter numbers 1-3");
scanf("%d", &no);
switch(no)
{
case 1:
printf("\n one");
break;
case 2:
printf("\n two");
case 3:
printf("\n three");
break;
default:
printf("\n Invalid Number");
}
getch();
}
```

Q.4(e) Write a program to exchange value of two integer numbers using function. [4]

(A)

```
#include<stdio.h>
#include<conio.h>
void exch(int a, int b);
void main( )
{
int a, b;
clrscr( );
printf("Enter value of a and b");
scanf("%d%d",&a,&b);
printf("\n Value before exchange:");
printf("\n a=%d\t b=%d",a,b);
exch(a,b);
getch( );
}
void exch(int a, int b)
{
int t;
t=a;
a=b;
b=t;
printf("\n Value after exchange:");
printf("\n a=%d \t b=%d",a,b);
}
```

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

(A)

```
#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 Four of the following: **[16]**

Q.5(a) Explain strlen( ) and strcat( ) string handling function from standard library. **[4]**

(A) **strlen( )**

This library function counts the length of the string.

Syntax : strlen(string1);

Example :

i=strlen(st 1);

strcat( )

This library function concatenates a string onto the end of the other string.

Syntax : strcat(string1, string2);

Example :

strcat(str1,str2);

Here, str2 will be concatenated at the end of the str 1.

Q.5(b) Write a program to sort array element in ascending order. **[4]**

(A)

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int arr[10];
int i,j,temp;
clrscr( );
for(i=0;i<10;i++)
{
scanf("%d",&arr[i]);
}
for(i=0;i<9;i++)
{
for(j=i+1;j<10;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<10;i++)
printf("%d",arr[i]);
getch( );
}
```

Q.5(c) What will be output of following c code? **[4]**

```
#include<stdio.h>
int main(){
    int i=1;
    for(i=0;i=-1;i=1) {
        printf("%d ",i);
        if(i!=1) break;
    }
    return 0;
}
```

(A) –1

Q.5(d) What is array of structure? List any two differences between array and array of structure. **[4]**

(A) 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];
}s[5];
```

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(e) State four storage class. Explain any one. **[4]**

(A) 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.5(f) Write the meaning of declaration: int * ptr;. **[4]**

(A)

```
main()
{
    int i = 3;
    printf("\n address of i = %u", &i);
    printf("\n Value of i = %d", i);
}
```

The o/p
Address of i = 65524
Value of i = 3

In the first printf statement '&' is used which is C's 'Address of'operator. The expression &i returns the address of the variable i. which in this case happens to be 65524. Since this number represents an address, there is no question of a sign being associated with it. Hence it is printed out using %u, which is a format specifier for printing an unsigned integer.

The other pointer operator available in C is "*", called 'value at address' operator. It gives the value stored at a particular address. The 'value at address' operator is also called as 'indirection' operator.

Q.6 Attempt any Four of the following: [16]

Q.6(a) Explain global and local variable with example. [4]

(A) **Global Variables** are declared before the main function. Global Variables can be accessed in any function in a program. Global Variables are alive till the end of the program.

Example of global variable

```
//program to find the sum of two numbers
#include<stdio.h>
int a,b,result;   //declaration of global variables
void main( )
{
…
}
```

Here a, b and result are global variables which are declared before the main function.

Local variables are declared inside a function. Local Variables cannot be accessed outside.

Example of local variable

```
//program to add any two integers
#include<stdio.h>
void main( )
{
int a,b,sum;      //declaration of local variable
…
}
```

Here a, b and sum are local variables which are declared in main function.

Q.6(b) Define pointer. How it declared and initialized? [4]

(A) A pointer is a variable which stores memory address of another variable which is of similar data type.

Declaration :
Syntax :

```
datatype *pointer_variable_name;
```

Example :

```
int *p;
```

Initialization :
Syntax :

```
pointer_variable_name = &variable_name;
```

Example :

```
int a;
int *p;
p=&a;
```

Q.6(c) Define : (i) Function definition (ii) Function body [4]
(iii) Function call (iv) Function Prototype

(A) **(i) Function Definition**

A function is a self-contained block of code that performs a specific task.

(ii) Function Body

It is defined as the code associated with a function name. It is written inside the block of function definition.

(iii) Function call

Function call is invoking a function wherever it is needed in the program. It can be done by writing name of the function ended with semicolon and with proper arguments given as per the function definition and prototype.

(iv) Function prototype

Function prototype is a function declaration before it is invoked.
It consists of function return type, function name, parameter list and terminating semicolon.

Example :

```
main()
{
void add(int,int); ----------→ function prototype
int a=5,b=6;
add(5,6); ----------→ function call
}
void add(int x, int y) ----------→ function definition
{
int z= x+y; ---→ function body
printf("sum=%d",z);
}
```

Q.6(d) State any two advantages and disadvantages of pointer. **[4]**

(A) Advantages :

(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

(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(e) State four arithmetic operations perform on pointer with example. **[4]**

(A)

```
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.

Q.6(f) Write a program to exchange values of two variables using pointer. [4]

(A)

```
#include<stdio.h>
#include<conio.h>
void main()
    {
        int a, b, *p1, *p2, temp;
        clrscr();
        printf("enter value of a");
        scanf("%d", &a);
        printf("enter value of b");
        scanf("%d", &b);
        printf("\nvalues before exchange are a=%d b=%d", a, b);
        p1=&a;
        p2=&b;
        temp=*p1;
        *p1=*p2;
        *p2=temp;
        printf("\nvalues after exchange are a=%d b=%d", a, b);
        getch();
    }
```

❑ ❑ ❑ ❑ ❑