Royal Education Society's

COCSIT LATUR

FACULTY OF COMPUTER STUDIES

[NEW CBCS PATTERN]

BCA First Year (Semester-I)

Class:BCAFY Subject:C programming

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Explain break and continue statement.

Ans:-The break statement is used for two different purpose it can be used to terminated a case in the switch statement that is to terminate the given loop immediately and it by pass the normal loop condition test.

we often come across situations where we want to jump out of a loop instantly, without wetting to get back to the conditional test, the keyword break allows us to do this. when break is encountered inside any loop, control automatically passes to the first statement after the loop. A break is usually associated.

Ex:-

```
Switch()
{
Case 1:
Break;
Case 2:
Break;
Default:
}
```

ex:- void main()

{

int I;

for(i=1;i<=10;i++)

{

if(i==5)

{

break;

}

printf(“%d”,i);

}

getch();

}

Continue Statement:-

This statement is used to when execution of further statement. This stops execution of next statement ?& transfer control to the start of loop for further execution of loop. This statement should be used only within the loop.

Ex:- program to determine continue statement.

Void main()

{

int i;

for(i=1;i<=10;i++)

{

if(i==5)

```
{

continue;

}

printf("\n%d",i);

getch();

}
```

in the above ex I is initialized to 1 then it will check (i==5) if it is true it will execute printf()

statement. After execution I is incremented by 1 & again executes printf() statement. This process

continues till the condition i<=10 becomes true.

b) Define array. Explain with suitable example how to declare and initialize array.

Ans:- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type. Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

```
type arrayName [ arraySize ];
```

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement −

```
double balance[10];
```

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

You can initialize an array in C either one by one or using a single statement as follows

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

c) Define variable. Explain the rules for constricting variables in C language.

Ans:-A variables is data name that may be used to store the data value these variables name are

giving to location in the memory of computer where we are storing the data there are some

rules available for creation and deceleration of variables.

Rules for Constructing Variable Name.

a) A variable name is any combination of 1 to 8 alphabets, digits or

underscores. Some compilers allow variable names whose length could

be up to 40 characters.

b) The first character in the variable name must be an alphabet.

c) No commas or blanks are allowed within a variable name.

d) No special symbol other than an underscore (as in gross_sal) can be used

in a variable name.

Ex:-intI,net_sal;

Float a;

d) Write a program in C to print the numbers from 4 to 9.

Ans:- #include<stdio.h>

#include<conio.h>

Void main()

{ int n;

Clrscr()

```
For(n=4;n<=9;n++)
Printf(“  %d”,n);
Getch();
}
```

Output:

4 5 6 7 8 9

e) Write a C program to swap two values.

Ans:-

```
#include<stdio.h>
#include<conio.h>
Void main()
{ int a,b,c;
Clrscr()
Printf(“enter the values for a and b”);
Scanf(“%d %d”,&a,&b);
c=a;
a=b;
b=c;
Printf(“a= %d b=%d”,a,b);
Getch();
}
```

Output:

10 20

A=20 b=10

f) Define string. How string is declared and initialized ?

Ans:- The string can be defined as the one-dimensional array of characters terminated by a null ('\0'). The character array or the string is used to manipulate text such as word or sentences. Each character in the array occupies one byte of memory, and the last character must always be 0. The termination character ('\0') is important in a string since it is the only way to identify where the string ends. When we define a string as char s[10], the character s[10] is implicitly initialized with the null in the memory

An array of characters (or) collection of characters is called a string.

Declaration

Refer to the declaration given below −

```
char stringname [size];
```

For example - char a[50]; a string of length 50 characters.

Initialization

The initialization is as follows −

- Using **single character** constant −

```
char string[20] = { ‘H’, ‘i’, ‘l’, ‘l’, ‘s’ ,‘\0’}
```

g) What is structure? Explain the C syntax of structure declaration with example

Ans:-

Structure in c is a user-defined data type that enables us to store the collection of different data types. Each element of a structure is called a member. Structures ca; simulate the use of classes and templates as it can store various information

The **,struct** keyword is used to define the structure. Let's see the syntax to define the structure in c.

```
1. struct structure_name
2. {
3.     data_type member1;
4.     data_type member2;
5.     .
```

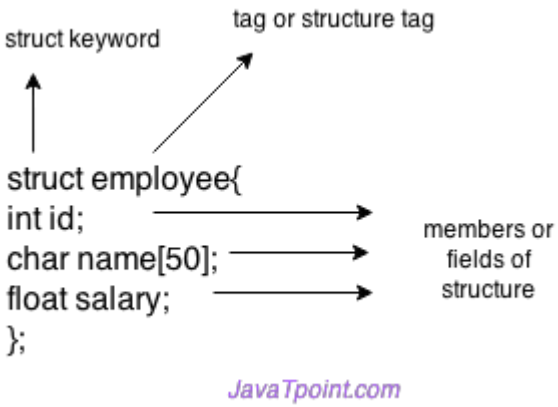
```
    .
    data_type memeberN;
};
```

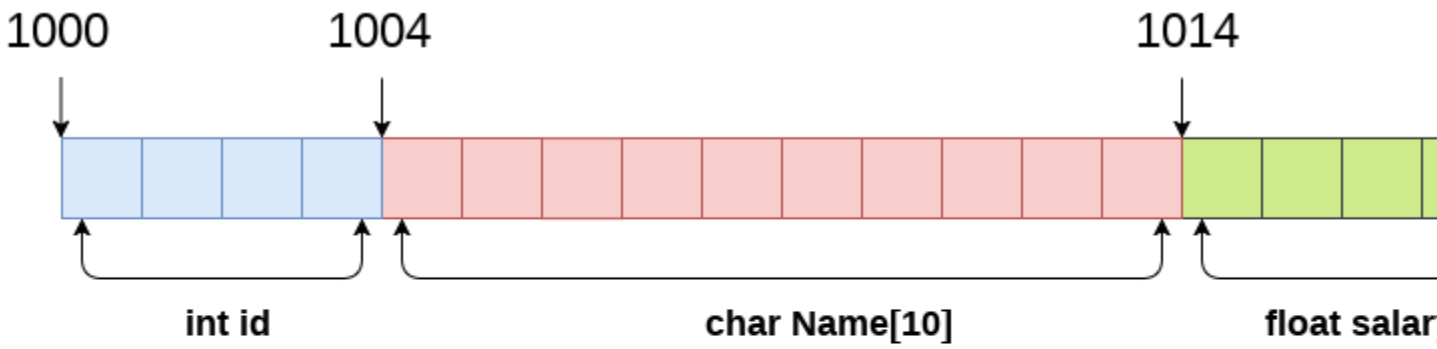
Let's see the example to define a structure for an entity employee in c.

```
struct employee
{   int id;
    char name[20];
    float salary;
};
```

The following image shows the memory allocation of the structure employee that is defined in the above example.

Here, **struct** is the keyword; **employee** is the name of the structure; **id**, **name**, and **salary** are the members or fields of the structure. Let's understand it by the diagram given below:






```
struct Employee
{
    int id;
    char Name[10];
    float salary;
} emp;
```

sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
sizeof (int) = 4 byte
sizeof (char) = 1 byte
sizeof (float) = 4 byte

Q. 2 Attempt any three of the following (5 Marks each) 15

A Explain structure of a C program .

Ans:-Structure of C Programming

1) Documentation Section

2) Header Section [link section]

3) Definition Section

4) Global Declaration

5) Main Section

6) Sub Program Section

1) Documentation Section

We can write title of program in that section using comments line

there are two types of comment line

a) Single line comment:-it is starting with // [double slash] this type of comment line are used for single line comment.

b) Multi line Comment /* */:-This type of comment line is used to give additional information about the program in two or more then two lines this type of comment line is started with /* & end with '*/' there is no effect on the program by using comment line.

2) Header Section[link section]

In this section we can use the Header file such as # 'stdio.h' &'conio.h' , string.h, such as Header file are used to give c linking support to the functions used in c program.

Stdio.h stands for slandered I/O file. Which support to standard I/O function as printf() &scanf().

Similarly conio.h is stand for console I/O hidder file which support to console I/O functions such as clrscr(); &getch().

3) Definition Section

In definition section we define constant value by using #defines ex:-#define

pie 3.14 In above example to define constant value 3.142 variable.

4) Global Declaration Section

There are some variables that are used in more than one function such variables are called global variable they are declared in the global declaration section that in outside of all function.

5) Main() Section

Every C program have only one main() function is start with ({) & close with (}) main function have main two part i.e declaration part & execution part.

a) Declaration part:-

In this part we declare the variables before using program. In declaration part we declare variables with their particular data types such as int, float, char etc.

b) Execution Part:-

In this part of main() section we use execution statements in main() section every statement & function are terminated by (;).

6) Sub program section:-

In sub program section we can use user defined function. User defined function are function which are defined by user.

Simple C Program

/* This is my first ‘c’ program */

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("Wel come C programming");
getch();
}
//Write a program two addition of two numbers
#include<stdio.h>
#include<conio.h>
Void main()
{
Clrscr()
Printf("The addition of two number is %d",20+30);
Getch();
}
```

b) What the formatted input and output functions. Explain with examples

Ans:-Formatted Function:-

printf():-

This function is used to display result on the screen it can be used to display any combination of numerical value as will as char or string. It requires conversion symbol and variable names to print the data. The conversion symbol and variable names should be same in number.

Syntax:-

printf(“<Format string>” ,arg1,arg2,....);

Ex:-

```
void main()
{
int a=3;
float b=5;
char c='b';
printf("%d %f %c",a,b,c);
}
```

//display the ascii value

```
#include<stdio.h>
#include<conio.h>
void main()
{
int y=65;
clrscr()
printf("%c %d",y,y);
getch();
}
```

scanf():-

The input data or information can be enter to the computer through a standard input device by using scanf() function. This function can be used to enter any combination of numerical value, single char, or strings this function return the number of data item that have been successful we can declare the scanf() function as follow.

scanf(“<Format string>” ,&arg1,&arg2,....);

In the format string various format specifies are used

Ex:-scanf(“%d”,&a);

The argument1,argument2,.....are the argument at that represent that indusial data item in the format string these argument are return of variables. The scanf() statement requires ‘&’ operator called address operator. The address operators print the memory location of the variable. Here in the scanf() statement the role of ‘&’ operator is to indicate the memory location of the variable. So that value read would be placed at that location.

Ex:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
inta,b,c;
clrscr();
scanf(“%d%d”,&a,&b);
c=a+b;
printf(“%d”,c);
getch();
}
```

c) Write a C program to find the area and perimeter of a rectangle

Ans:-

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

int main(){
```

```
    float length, width, area;
    printf("Enter length of Rectangle\n");
    scanf("%f", &length);
    printf("Enter width of Rectangle\n");
    scanf("%f", &width);
    /* Area of Rectangle = Length X Width */
    area = length * width;
 perimeter = 2*(length + width);
 printf("Perimeter of Rectangle : %0.4f\n", perimeter);

    printf("Area of Rectangle : %0.4f\n", area);

    getch();
    return 0;
}
```

Program Output

```
Enter length of Rectangle
6.5
Enter width of Rectangle
5
Area of Rectangle : 32.5000
Perimeter of Rectangle : 23.0000
```

d) Explain switch statement with syntax and example

Ans:- By using switch statement, which have multiple collection but (alternative increases)

complexity increase and program become easy to read and debug. Switch statement is multi-case statement. The keyword switch statement value of condition match with label

Syntax:

```
switch(expression)
{
case value 1:
statement 1;
break;
case value 2:
statement 2;
break;
Default:
Default block;

}
```

If condition is doesn't match with label of case (1) than it checks condition value is match with label of case (2). If that value is match than execute statement block of case (2) and control transfer to statement x and so on---

We have to remember that, case labels are single character constants.

Explanation of switch statement

Expression: it is an integer expression or character.

Value1, value 2 : this are constants or constants expression known as case label.

Block 1, block 2: this are statement which may content one or more statement.

Case label must be end with colon(:).

default: the default is an optional case which it will execute expression value, doesn't match

with any case value.

 In a switch there can be either variable or expression

 If it is a variable it must be either integer or character

 If it is an expression it must be an arithmetic expression.

 There can be any number of cases.

 Every case should have an unique value.

 These cases can be written in any sequence.

 In a case there can be any number of statement.

 It is possible to have the nested switch.

 After every case break statement is compulsory.

 Default is a case which gets selected when none of the case value matches with

the result of expression.

 Default case is optional.

Write a program that reads a number between 1 to 7 and display the day name

```
main()
{
Int day;
Printf(“Enter a number between 1 to 7 \n”);
Scanf(“%d”,&day);
Switch(day)
{
Case 1:printf(“Monday\n”);
Break;
Case 2:printf(“Tuesday\n”);
```

```
Break;

Case 3:printf(“Wednesday\n”);

Break;

Case 4:printf(“Thursday\n”);

Break;

Case 5:printf(“Friday\n”);

Break;

Case 6:printf(“Saturday\n”);

Break;

Case 7:printf(“Sunday\n”);

Break;

Default:printf(“Monday\n”);

}
```

e) WAP to display 3*3 matrix addition.

Ans:-

```
/ADDITION OF TWO MATRICES.
#include<stdio.h>
#include<conio.h>
void main()
{
int m1[3][3],m2[3][3],add[3][3],i,j;
clrscr();
printf("Enter first matrix elements(any 9 numbers)\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
```

```
scanf("%d",&m1[i][j]);
}
}
printf("Enter second matrix elements(any 9 numbers)\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&m2[i][j]);
}
}
printf("\n**********************************************************\n");
printf("Resultant matrix is as follows\n");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
add[i][j]=m1[i][j]+m2[i][j];
printf("%d ",add[i][j]);
}
printf("\n");
}
getch();
}
```

Output:-

1 2 3 4 5 6 7 8 9

9 8 7 6 5 4 3 2 1

10 10 10 10 10 10 10 10 10

Q. 3 Attempt any three of the following (5 Marks each) 15

a) Explain the following operators in C language

i) Relational ii) Logical iii) Conditional

Relational operators

If we want to compare two values then Relational operators are used C supports six

Relational operators

1 >

2 >=

3 <

4 <=

5 = =

6 !=

The simple format of Relational operators as follows

Expression Relational operator's expression

The value of real expression is dissention statement if

//Write a program to use various relational operators and display their return values.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf("\nCondition : Return Values \n");
printf("\n10!=10 : %5d",10!=10);
printf("\n10==10 : %5d",10==10);
printf("\n10>=10 : %5d",10>=10);
printf("\n10<=10 : %5d",10<=10);
printf("\n10!=9 : %5d",10!=9);
getch();
}
```

logical operators

C language support three logical operators

&& Logical And

|| Logical or

! Logical Not

a<b && x==30

a>b || b>c

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Conditional operators

Expirations ?expression2 : expression3

? :

a=3;

b=4;

x=(a<b)? a:b;

//write a program to find out the given number greater or not by using

conditional operators.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
Inta,b;
Printf(“Enter two number”);
Scanf(“%d%d”,&a,&b);
a>b?printf(“A is greater number”):Printf(“B is greater number”);
getch();
}
```

in above expression a is less then b if this condition is true then value of a will be to assign to the expression and if given condition is false then value of b it will be assign to the expression

b) Explain different data types used in C

Ans:-C language is rich in data type like other languages ‘C’ language has also it’s own data types data types are used to define or declare the type [data type] of particular variable, constant, function etc.

In C language there are three classes of data types.

1) Primary data type.

2) Derived data type.

3) User-defined data types.

Data type are always written in small letters each data types has it's own feature depending upon a condition the particular data types is used in the name of data type we cannot use in valid space.

1) Primary Data Type:- it is also known as standard of or built in data type all compiler support fundamental data type. Primary data type is divided into 3 type.

Int [integer]:-

Integer are defined by keyword 'int' integer are store for no we cannot store decimal point no in that data type integer occupies one word of storage i.e 2 bytes or 16 bits. The range of integer data type is -32768 to 32767 'C' has support to classes of integer storage namely intshorint& long int variable than they increases the range of value aswill as size of it. 32 bit word length can store an integer ranging from -2,147,483,648to 2147483647.

Ex:-Int a=5; int b=6;

Float

The real no are known as floating value. These no contain functions part. This is defined by keyword 'float' float value requires 4 bytes or 32 beats. The range of float is 3.4e-38 to 3.4e38.

Ex:- float pie=3.14;

character (char)

Generally those data types are used when we have to storage a single character or string to variable. It requires 1 byte or 8 beats to store in memory. It is defined by keyword as 'char' the range of char is -128 to 127.

Ex:-char name;

Name='a';

Char name[15]="Bhosle";

Double

It is very similar to float data type the major difference between float & double data type is size & range of the double data type is always greater than float data type size of double data type is 64 beats, 8 bytes range of double data type is 1.7e-308 to 1.7e +308 .

Void Type

The Void type has no value. This is usually used to specify the type of functions. The type of a function is said to be void when it does not return any value to the calling function. It can also play the role of a generic type, meaning that it can represent any of the other standard types.

User defined data type

These data type are defined by using keyword 'typedef' it takes following format.

Syntax :-typedef type identifier name

c) Write a C program that computes the size of int, float, and char

Ans:- #include <stdio.h>

```
int main()
{   printf("Size of a Character = %ld Byte", sizeof(char));
    printf("\nSize of an Integer  = %ld Byte", sizeof(int));
printf("\nSize of a Float     = %ld Byte", sizeof(float));
  return 0;}
```

Output:

Size of a Charcter=1 Byte

Size of an Integer=2 Byte

Size of a Float=4 Byte

d) Explain Nested if-else statement with example.

Ans:-Nested if –else statement

It is perfectly all right if we write an entire if-else construct within either the body of the if statement or the body of an else statement. This is called as nesting of if. In this kind of statement no of logical condition are checked. For executing various statement. Nested if-else can be chained with one another If the condition is check if condition is false then control passes to else statement block where condition is again checked with if statement. This process continue become condition is true if condition is true then execute the true statement block otherwise last else statement block will be execute.

Syntax:-

```
if(Condition 1)
{
if(Condition 2)
{
Body of True Statement Block
}
else
{
Body of False Statement Block
}
}
```

```
else
{
if(Condition 3)
{
Body of True Statement Block
}
else
{
Body of False Statement Block
}
}
```

Ex-

```
if(a>b)

{
if(a>c
{
printf(“a is grater”);
}
else
{
printf(c is grater”);
```

```
}
}

else
{
if(b>c)
{
printf(“B is grater”);
}
else{
printf(“c is grater”);
}
}
```

In that first check the condition. If the given condition1 is true then again check the condition 2 if the given condition 2 is true then execute the true statement block and control

Simple If ..else statement flow chart

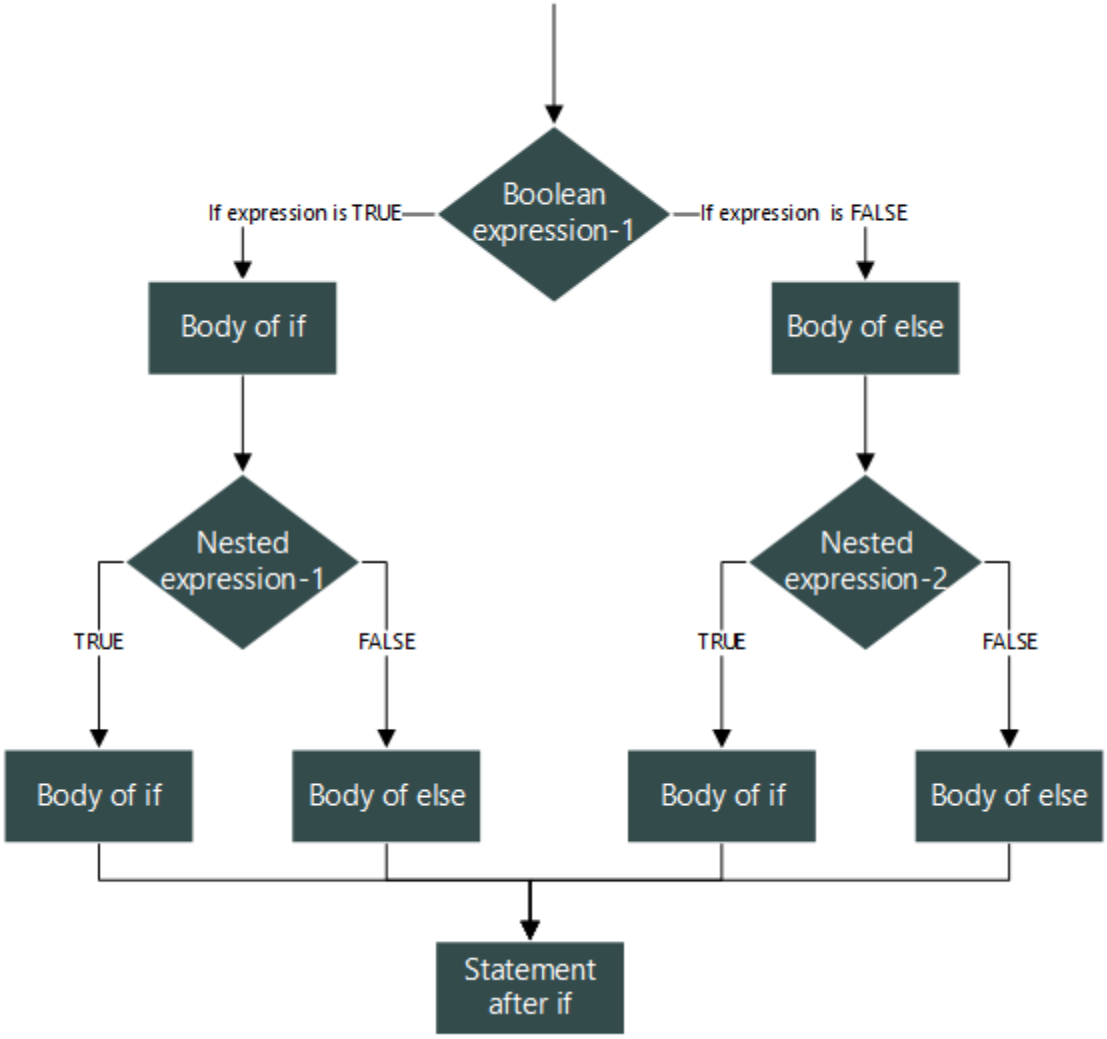
Ex-

```
if(a>b)

{
if(a>c
{
printf(“a is grater”);
```

```
}
else
{
printf(c is grater”);
}
}

else
{
if(b>c)
{
printf(“B is grater”);
}
else{
printf(“c is grater”);
}
}
```

In that first check the condition. If the given condition1 is true then again check the condition 2 if the given condition 2 is true then execute the true statement block and control

Simple If ..else statement flow chart



transfer to statement x. if the given condition 2 is false then without executing the true statement block execute the false statement block of condition 2 and control transfer to statement x. if given condition 1 is false then control transfer to else statement again check the condition 3 if the condition 3 is true then execute the true statement and control transfer to statement x. if the given condition 3 is false then execute the false statement block and control transfer to statement x.

When this structure gets executed and as soon as any one of the statement gets

executed, immediately the chain breaks & computer goes next statement. There are many

possible forms of nesting. It always depends on our problem.

e) Write a C program to find the biggest of three numbers

Ans:- **Alorithm:**

1. Take the three numbers as input and store them in variables.
2. Check the first number if it is greater than other two.
3. Repeat the step 2 for other two numbers.
4. Print the number which is greater among all and exit.

```c
#include <stdio.h>

int main()
{
    int a, b, c;
    printf("Enter three numbers: \na: ");
    scanf("%d", &a);
    printf("b: ");
    scanf("%d", &b);
    printf("c: ");
    scanf("%d", &c);
    if (a > b && a > c)
        printf("Biggest number is %d", a);
    if (b > a && b > c)
        printf("Biggest number is %d", b);
    if (c > a && c > b)
        printf("Biggest number is %d", c);
    return 0;
}
```

Output—

```
Enter three numbers:
a: 6
b: 8
c: 10
Biggest number is 10
```

Q. 4 Attempt any three of the following (5 Marks each) 15

a) List the differences between while loop and do-while loop.

Ans:-

SR.NO	while loop	do-while loop
1.	While the loop is an entry control loop because firstly, the condition is checked, then the loop's body is executed.	The do-while loop is an exit control loop becau in this, first of all, the body of the loop is execut then the condition is checked true or false.
2.	The statement of while loop may not be executed at all.	The statement of the do-while loop must executed at least once.
3.	The while loop terminates when the condition becomes false.	As long as the condition is true, the compil keeps executing the loop in the do-while loop.
4.	In a while loop, the test condition variable must be initialized first to check the test condition in the loop.	In a do-while loop, the variable of test conditi Initialized in the loop also.
5.	In a while loop, at the end of the condition, there is no semicolon. **Syntax:** `while (condition)`	In this, at the end of the condition, there is semicolon. **Syntax:** `while (condition);`
6.	While loop is not used for creating menu-driven programs.	It is mostly used for creating menu-drive programs because at least one time; the loop executed whether the condition is true or false.
7.	In a while loop, the number of executions depends on the condition defined in the while block.	In a do-while loop, irrespective of the conditi mentioned, a minimum of 1 execution occurs.

8.	**Syntax of while loop:**
	`while (condition)`
	`{`
	`Block of statements;`
	`}`
	`Statement-x;`
		Syntax of do-while loop:
		`do`
		`{`
		`statements;`
		`}`
		`while (condition);`
		`Statement-x;`

9.	**Program of while loop:**	**Program of do-while loop:**

	Program of while loop:
	
	#include
	#include
	Void main()
	{
	int i;
	clrscr();
	i = 1;
	while(i<=10)
	{
	printf("hello");
	i = i + 1;
	}
	getch();
	}
		#include
		#include
		Void main()
		{
		int i;
		clrscr();
		i = 1;
		do
		{
		printf("hello");
		i = i + 1;
		}
		while(i<=10);
		getch();
		}

b) Write a C program to find the factorial of a number .

Ans:-
```
#include<stdio.h>
int main()
{
 int i,fact=1,number;
 printf("Enter a number: ");
  scanf("%d",&number);
    for(i=1;i<=number;i++){
      fact=fact*i;
  }
  printf("Factorial of %d is: %d",number,fact);
return 0;
}
```

Output:

```
Enter a number: 5
Factorial of 5 is: 120
```

c) Write a C program to find the largest element in an array

Ans:- ALGORITHM:

- **STEP 1:** START
- **STEP 2:** INITIALIZE arr[] = {25, 11, 7, 75, 56}
- **STEP 3:** length= sizeof(arr)/sizeof(arr[0])
- **STEP 4:** max = arr[0]
- **STEP 5:** SET i=0. REPEAT STEP 6 and STEP 7 i<length
- **STEP 6:** if(arr[i]>max)
  max=arr[i]
- **STEP 7:** i=i+1.
- **STEP 8:** PRINT "Largest element in given array:" assigning max.
- **STEP 9:** RETURN 0
- **STEP 9:** END.
- **Program:**

```
#include <stdio.h>

int main()
{
    //Initialize array
    int arr[] = {25, 11, 7, 75, 56};

    //Calculate length of array arr
    int length = sizeof(arr)/sizeof(arr[0]);

    //Initialize max with first element of array.
    int max = arr[0];

    //Loop through the array
    for (int i = 0; i < length; i++) {
        //Compare elements of array with max
       if(arr[i] > max)
           max = arr[i];
    }
    printf("Largest element present in given array: %d\n", max);
```

- `return 0;`
- `}`

Output:

```
Largest element present in given array: 75
```

d) Write a C program to check whether the given number is prime or not

Ans:-

```c
#include<stdio.h>


int main()
{
    int num, count, prime = 1;

    printf("Enter a positive number\n");
    scanf("%d", &num);

    for(count = 2; count < num; count++)
    {
        if(num % count == 0)
        {
            prime = 0;
            break;
        }
    }

    if(prime)
```

```
        printf("%d is a Prime Number\n", num);
    else
        printf("%d is a not Prime Number\n", num);

    return 0;  }
```

Output :

Enter a number
7
7 is prime number

e) Explain for loop with example.

Ans:-For loop

This statement is used when calculations are to be carried out between initial and final value with step. The for loop condition consist of three actions namely initialization, testing and incrementing/ decrementing. Each action should be separated by semicolon(;) also this loop statement is used when number of execution time is known in advance.

It is an entry control loop having syntax as follow.

Syntax-

for(initial value;condition;increment/decrement)

{

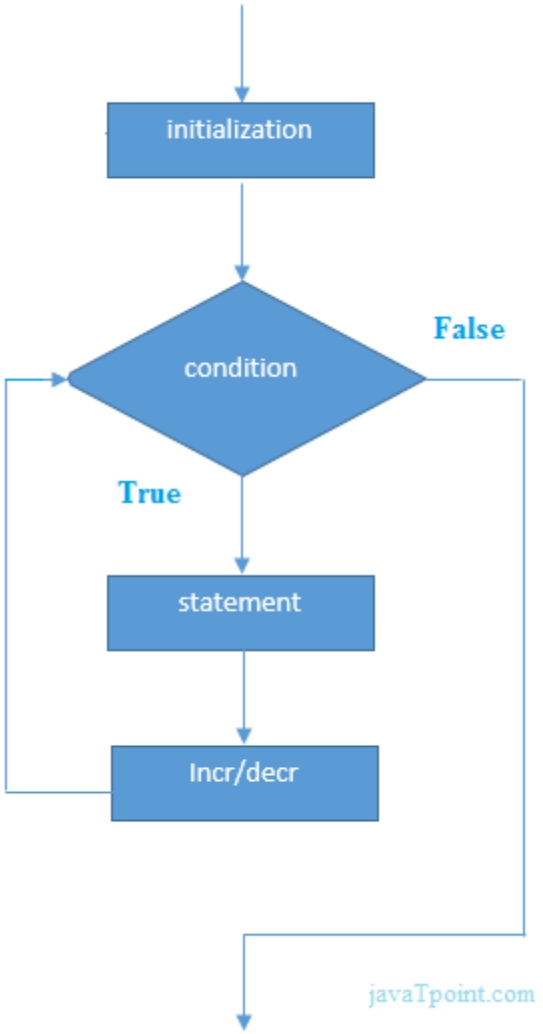
body of for loop;

}

In this loop the initial value, condition, increment/decrement is specified in the same line. Here computer starts with the initial value, checks the condition. If the condition is true computer

executes the body and automatically goes up, it takes the increment or decrement automatically and it continues. Remember, if the body of for continue a single statement, no need of brackets. But if there are more than one statement in the body then they must be enclosed between brackets. Here every thing is specified in the same line and hence we can tell the number of repetitions it can take. When we know the number of repetitions in advance than we should go for the for loop.

Flowchart of for loop in C



Ex:

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

Q .5 Short notes on any three of the following (5 Marks each) 15

a) Keywords

Keywords are the words whose meaning has already been explained to the C compiler. The keywords cannot be used as variable names because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer.

Auto,Break Case Char Const Continue

Default do Double Else Enum extern

Float far For Goto If Int

Long near Register Return Short Signed

Static struct Switch Typedef Union Unsigned

Void while

There are 32 keywords used in C all keywords must be written in lower case.

Some compilers may also include some or all of the following keywords.

Ada Far Near AsmFortran Pascal

Entry Huge

b) Union

Ans:-**Union** can be defined as a user-defined data type which is a collection of different variables of different data types in the same memory location. The union can also be defined as many members, but only one member can contain a value at a particular point in time.

Union is a user-defined data type, but unlike structures, they share the same memory location.

Let's understand this through an example.

```
1. struct abc
2. {
3.    int a;
4.    char b;
5. }
```

The above code is the user-defined structure that consists of two members, i.e., 'a' of type **int** and 'b' of type **character**. When we check the addresses of 'a' and 'b', we found that their addresses are different. Therefore, we conclude that the members in the structure do not share the same memory location.

When we define the union, then we found that union is defined in the same way as the structure is defined but the difference is that union keyword is used for defining the union data type, whereas the struct keyword is used for defining the structure. The union contains the data members, i.e., 'a' and 'b', when we check the addresses of both the variables then we found that both have the same addresses. It means that the union members share the same memory location.

c) printf and scanf statements

Ans:- printf():-

This function is used to display result on the screen it can be used to display any combination of numerical value as will as char or string. It requires conversion symbol and variable names to print the data. The conversion symbol and variable names should be same in number.

Syntax:-

printf("<Format string>" ,arg1,arg2,....);

Ex:-

```
void main()
{
int a=3;
float b=5;
char c='b';
printf("%d %f %c",a,b,c);
}
```

//display the ascii value

```
#include<stdio.h>
#include<conio.h>
void main()
{
int y=65;
clrscr()
printf("%c %d",y,y);
getch();
}
```

scanf():-

The input data or information can be enter to the computer through a standard input device by using scanf() function. This function can be used to enter any combination of numerical value, single char, or strings this function return the number of data item that have been successful we can declare the scanf() function as follow.

scanf("<Format string>" ,&arg1,&arg2,....);

In the format string various format specifies are used

Ex:-scanf("%d",&a);

The argument1,argument2,.....are the argument at that represent that indusial data item in the format string these argument are return of variables. The scanf() statement requires '&' operator called address operator. The address operators print the memory location of the variable. Here in the scanf() statement the role of '&' operator is to indicate the memory location of the variable. So that value read would be placed at that location.

Ex:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
inta,b,c;
clrscr();
scanf("%d%d",&a,&b);
c=a+b;
printf("%d",c);
getch();
}
```

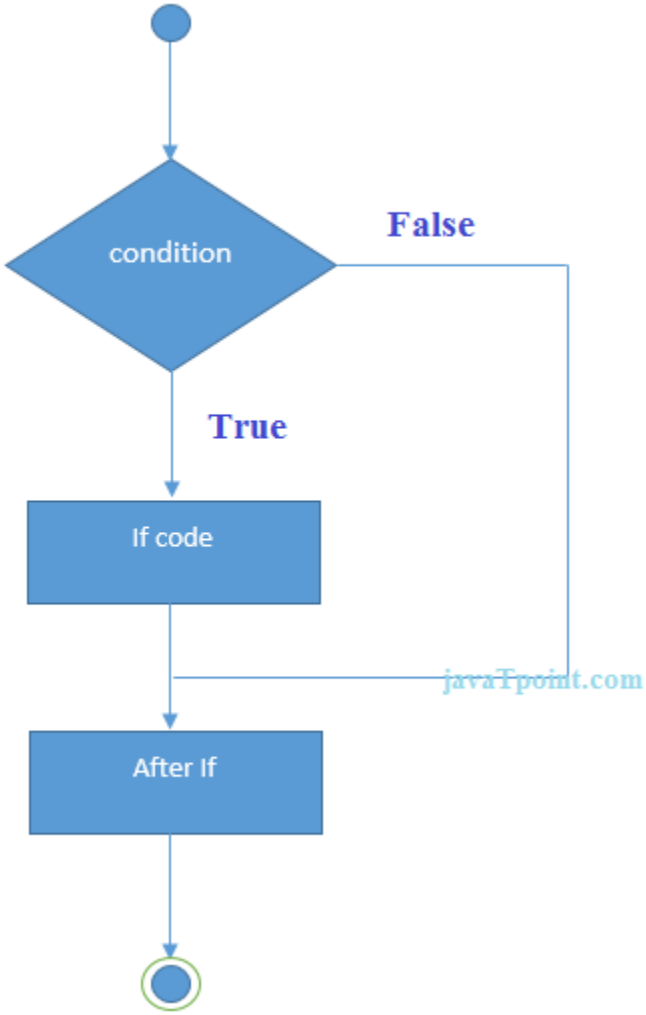
d) If-else statements

Ans:-If – else statement

The if – else statement is an extension of the simple if statement. In this statement first check the condition if condition is true then body of if (true) statement is executed otherwise body of else (false) statement is executed. In this statement either body of if or body of else is executed and then only the control moves to the next statement.

Syntax:-

```
if (condition)
{
Body of true statement block;
}
else
{
Body of false statement block;
}
```



Ex:-

```
if (a>b)
{
Printf("\n A is grater then B");
}
Else
{
```

```
Printf(“\nA is less then B”);
}
```

This statement provides the alternates in both the situations, if condition is true the body of if is executed if condition is false the body of else is executed.

Here computer can not proceed without executing either body of if or body of else.

Ex:

```
void main()
{ inta,b;
printf(“\nEnter the Number a & b”);
scanf(“%d%d”,&a,&b);
if(a>b)
{
printf(“A is grater number”);
}
else
{
printf(“B is grater number”);
}getch();
}
```

e) goto statement

ans:-Goto Statement:

This statement transfers the control from one statement to other statement in the program which may not be in the sequence. The general form of the goto statement is

goto label;

----

label:

towards jump

label: statement

----

goto label;

backward jump

wheregoto requires a level in order to identify the place where the branch is to be made label.

It is an valid variable name & must be followed by a (:) the program either before or after the goto label statement as shown below.

Forward jump:

If the label is placed after goto label statement will be skipped such a jump is known as forward jump.

Backward jump:

If the label is before the goto statement a loop will form & some statement will be executed repeatedly such a jump is known as backward jump.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main ()
```

```
{
inta,b;
clrscr();
xyz:printf("enter the number");
scanf("%d",&a);
if(a%2==0)
{
printf("given num is even ");
}
else
{
printf("\nthe given num is odd");
}
printf("\nif you want to continue");
printf("\n Enter 6 to yes 7 to no");
scanf("%d",&b);
if(b==6) goto xyz;
if(b==7) exit(0);
getch();}
```