CHANDIGARH ENGINEERING COLLEGE, LANDRAN (MOHALI)

DEPARTMENT OF APPLIED SCIENCES

PPS (BTPS-101-18) **Branch-ECE/ME Sem-1**

B.Tech -Question Paper-3 Solution
Odd Sem -2019-2020- QP

Section-A(10*2= 20 Marks)

Write briefly:

1. **What is the syntax of nested if statement?**

Ans:

Syntax of nested if statement:

```
if(condition1)
{
  if(condition2)
        //statements to be executed if condition1 and 2 is true
}
else
{//statements to be executed if condition1 is false}
```

Example:

```
 if(a>b)
{
 if(a>c)
 printf("a is greatest");
 else
    printf("c is greatest");
}
 else
{  if(b>c)
     printf("b is greatest");
   else
      printf("c is greatest");
}
```

2. **What is a conditional operator in C?**

Ans:

Conditional operators return one value if condition is true and returns another value is condition is false. This operator is also called as ternary operator has three operands.

Syntax : (Condition? true_value: false_value);

Example : (A > 100 ? 0 : 1);

If A is greater than 100, 0 is returned else 1 is returned. This is equal to if else conditional statements

3. What is a case statement?

Ans:

When number of conditions (multiple conditions) occurs in a problem, either ladder if statement can be used or switch statement can be used. In switch statement to mention different options, case statement is used.

Example:

```
switch(A)
{
 case 1:
  printf("First option");
  break;
case 2:
  printf("Second option");
  break;
default:
 printf"Invalid Option");
}
```

4. What are Arithmetic operators?

Ans:

An arithmetic operator performs mathematical operations such as addition, subtraction and multiplication on numerical values (constants and variables).

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division(modulo division)

5. Differentiate compiler and interpreter?

Interpreter	Compiler
Translates one statement in a program at a time.	Scans the entire program and translates it as a whole into machine code.
Interpreters usually take less amount of time to analyze the source code. The overall execution time is less than compilers.	Compilers usually take a large amount of time to analyze the source code. The overall execution time is faster than interpreters.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.

6. What is a function?

Ans:

A Function is a self-contained block of statements that can be executed repeatedly whenever required. A function is group of statements that together perform a task. Every C program has atleast one function which is main().

7. How many bytes are required for int a[20] statement ?

Ans:

Int requires 4 bytes. So 20*4=80 bytes are reserved.

8. What is an algorithm?

Ans:

Algorithm means the logic of a program. It is a step-by-step description of how to arrive at a solution of a given problem.

9. How structure is represented?

Ans: A structure is a user defined data type in C.

'struct' keyword is used to create a structure.

Example:

```
struct addrress
{
  char name[50];
  char street[100];
   int pin;
};
```

10. What is a conditional statement?

Ans:

Conditional statements in C programming language are

1. if statement
2. if-else statement

3. ternary statement or ternary operator
4. nested if-else statement
5. switch statement

Total 5 – Questions to be attended opting at least two from section B and C (5*8=40 Marks)

Section- B (8 marks Questions)

11. Write a program to find difference of two matrix.

Matrix Subtraction

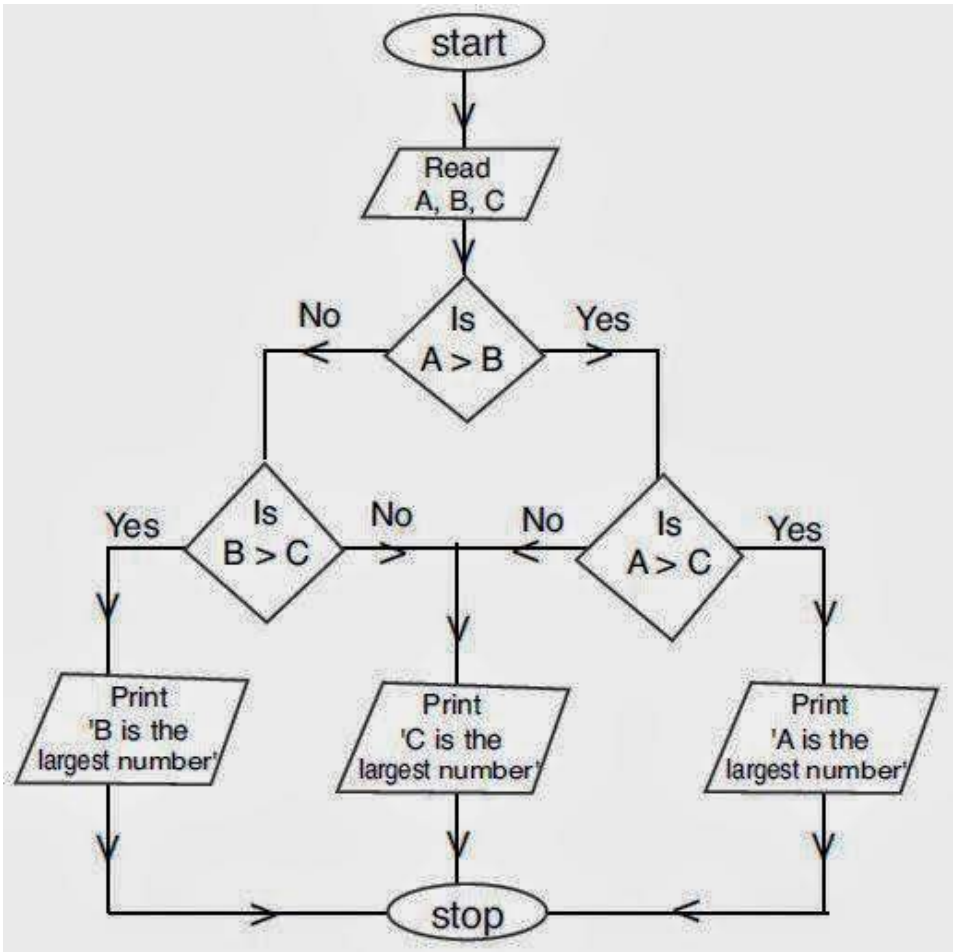
```
#include <stdio.h>
    int main()
{
  int m, n, c, d, first[10][10], second[10][10], diff[10][10];
 printf("Enter the number of rows and columns of matrix\n");
  scanf("%d%d", &m, &n);
  printf("Enter the elements of first matrix\n");
  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      scanf("%d", &first[c][d]);
 printf("Enter the elements of second matrix\n");
 for (c = 0; c < m; c++)
    for (d = 0 ; d < n; d++)
      scanf("%d", &second[c][d]);
 printf("Sum of entered matrices:-\n");
 for (c = 0; c < m; c++)
 {
    for (d = 0 ; d < n; d++)
   {
      diff[c][d] = first[c][d] - second[c][d];
       printf("%d\t", diff[c][d]);
   }
    printf("\n");
 }
 return 0;
}
```

Output:

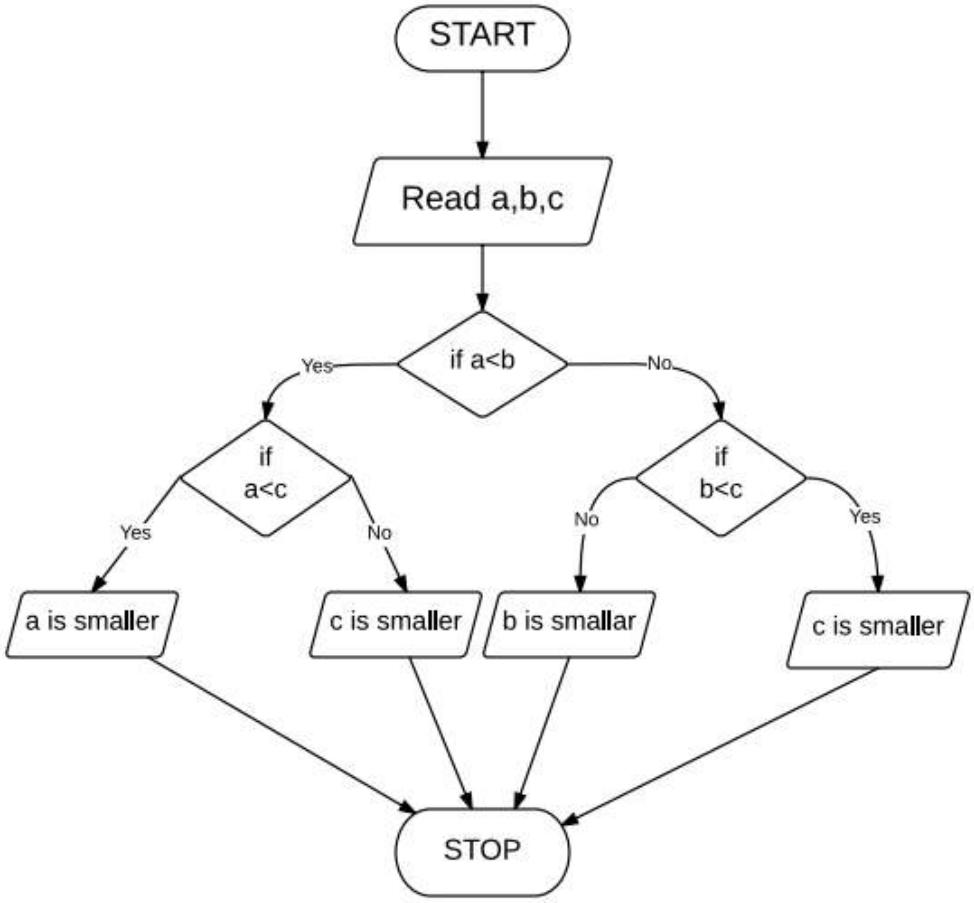
Enter the number of rows and columns of matrix
2 2
Enter the elements of first matrix
5 6
3 4
Enter the elements of second matrix
3 2
2 1
Difference of entered matrices:-
2 4
1 3

12. Draw the flowchart to find the largest and smallest of three numbers.

Greatest of three numbers



Smallest of three numbers:



13. Write a program to sort a given list of numbers in ascending order.

```
#include <stdio.h>
 int main()
{
  int n,A[1000],i,j,key;
   printf("Enter number of elements\n");
  scanf("%d", &n);
   printf("Enter %d integers\n", n);
   for (i=0;i< n;i++) {
    scanf("%d", &A[i]);
  }
```

```
 for (i=1;i < n ; i++)
{
 key=A[ i ];
j=i-1;
while(j > =0 && A [ j ] > key)
{
A[j+1]=A[j];
 j--;
 }
A [ j+1 ] = key;
}
 printf("Sorted list in ascending order:\n");

 for (i= 0;i<n;i++) {
   printf("%d\n",A[i]);
 }
 return 0;
}
```

Output:

Enter number of elements
5
Enter 5 integers
4
2
6
9
5
Sorted list in ascending order:
2
4
5
6
9

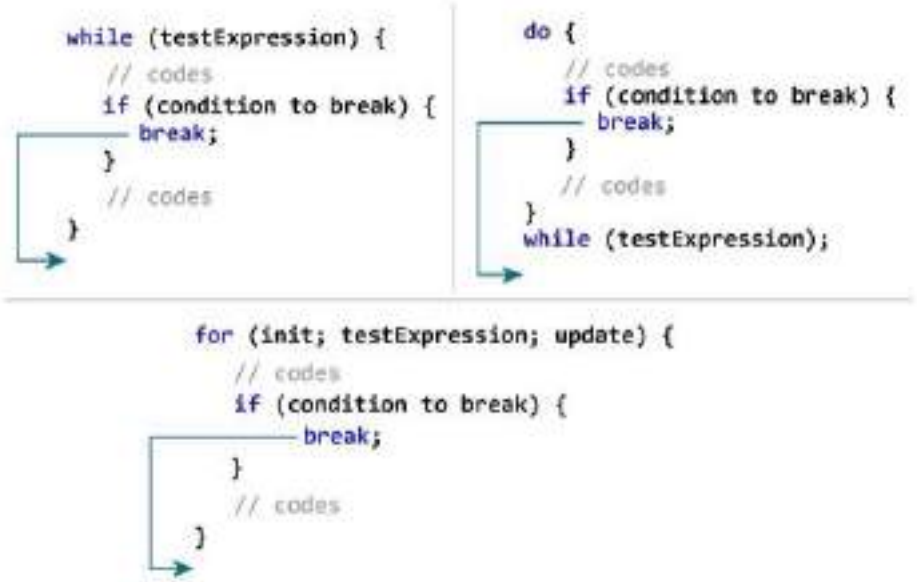
14. Write short notes on the following:

a)Break and Continue

Ans:Break:

The break statement ends the loop immediately when it is encountered. Its syntax is:
break;
The break statement is almost always used with if...else statement inside the loop.

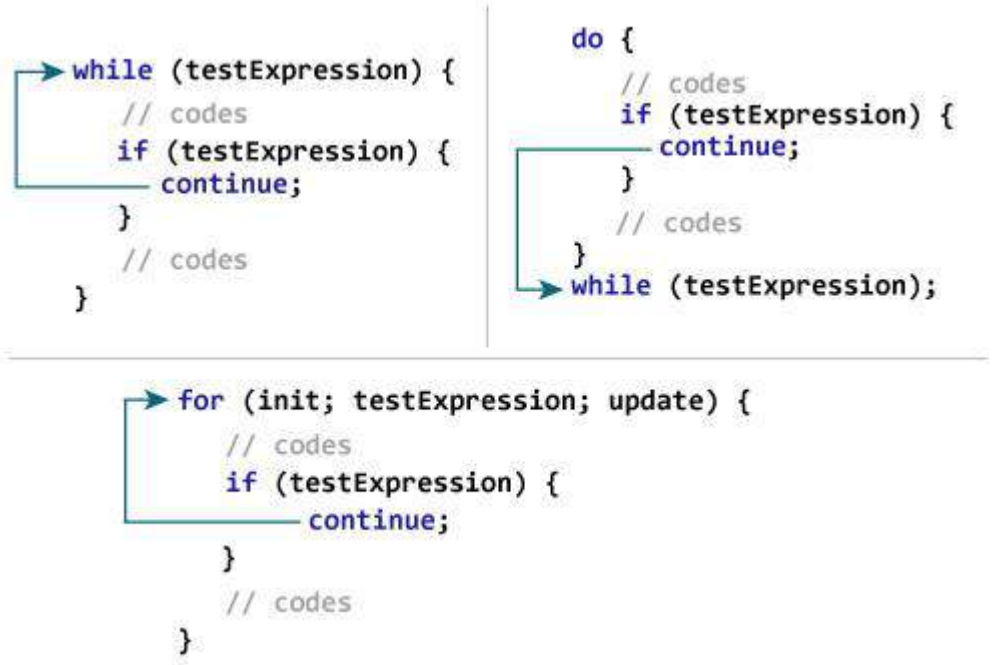
Example:



Continue:The continue statement skips the current iteration of the loop and continues with the next iteration. Its syntax is:

continue;

Example:



b)passing value between functions C

The call by value method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument. By default, C programming uses call by value to pass arguments. In general, it means the code within a function cannot alter the arguments used to call the function.

Consider the function swap() by passing actual values as in the following example −

```
#include <stdio.h>
 /* function declaration */
void swap(int x, int y);
 int main () {
   /* local variable definition */
   int a = 100;
   int b = 200;
    printf("Before swap, value of a : %d\n", a );
   printf("Before swap, value of b : %d\n", b );
    /* calling a function to swap the values */
   swap(a, b);
    printf("After swap, value of a : %d\n", a );
   printf("After swap, value of b : %d\n", b );
    return 0;
}
void swap(int x, int y)
{
   int temp;
   temp = x; /* save the value of x */
   x = y;    /* put y into x */
   y = temp; /* put temp into y */
     return;
}
```

Output:

```
Before swap, value of a : 100
Before swap, value of b : 200
After swap, value of a : 100
After swap, value of b : 200
```

It shows that there are no changes in the values, though they had been changed inside the function

Section –C (8 marks Questions)

15. Explain the following terms of C language with examples.

a) Static variables

b) External variables

c) Automatic variables

a) **Static Variable**:

Variables may be internal or external depending on the place where they are declared. If declared outside main, they are static global. If declared inside main, they are static local. When static, its garbage value is removed and initialized as NULL and then the value remains same throughout the execution.

Eg:

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   int x;
 static int y;
\printf("x=%d,y=%d",x,y);
}
```

Output:

```
X=1026   //garbage value
  Y=0
```

b)**External Variables(Global Variables):**

Variables which are available to all function are called external variables.They are declared outside the function body. If both global and auto variables have the same name in a program, first priority will be given to auto variables and external variable is hidden.

Eg:

```
   #include<stdio.h>
  #include<conio.h>
void block1();
 void block2();

   void main()
{
     block1();
```

```
    block2();
   printf("n=%d",n);
}
 void block1()
{
       printf("n=%d",n);
}
 void block2()
{
  printf("n=%d",n);
}
```

Output:

```
 n=10
 n=10
n=10
```

c) **Automatic Variables:**

Variables inside a function is called automatic variable. The variables by default is auto variable. The scope of the variables are local to the block in which they are defined. Once executed the contents and existence of the automatic variable gets vanished.

Eg:

```
#include<stdio.h>
#include<conio.h>
 void block1();
 void block2();
 void main()
{
  int n=10;
block1();
block2();
printf("n=%d", n);
}
 void block1()
{
   int n=20;
    printf("n=%d",n);
}
 void block2()
{
```

```
   int n=30;
   printf(“n=%d”,n);
}
```

Output:

```
  n=20
  n=30
 n=10
```

16. Write a recursive and non-recursive function or determining the factorial of a given number.

a) using recursive function.

```
#include<stdio.h>
#include<conio.h>
int fact( int);                          // function declaration
int main( )
{
int no,result;
clrscr( );
printf("Enter the required number:");
scanf("%d", &no);
result = fact( no);
printf("\n %d Factorial is : %d", no, result);
getch( );
return 0;
}
int fact(int n)
{
int ft,
if( n==1)
return 1;
else
ft= n*fact (n-1);
return ft;
}
```

b)without using recursive function.

```
#include<stdio.h>
#include<conio.h>
int fact( int);                          // function declaration
```

```
int main( )
{
int no,result;
clrscr( );
printf("Enter the required number:");
scanf("%d", &no);
result = fact( no);
printf("\n %d Factorial is : %d", no, result);
getch( );
return 0;
}
int fact(int n)
{
int ft,
for( ft=1; n>=1; n--)
ft=ft*n;
return ft;
}
```

Output:- 4 Factorial is : 24

17. Write the syntax with two different examples each for :
a)Struct
b)Union
State how much memory would be required for each of your example.

Structure is an userdefined data type. The memory that is allocated for a structure element is to store all the data fields which included in struct. The syntax is as follows

```
struct <identifier>
{
datatype1 element1;
datatype2 element2;
};
```

Example:

```
struct example1
{
  int a; // 4 bytes
  long b; // 8 bytes
  char c; // 1 byte
  double d; // 8 bytes
```

};
Here for example1 21 bytes are required and almost 24 bytes are reserved including data alignment.

Union is another important composite data structure in C programming language, it is similar to struct, except the memory allocation strategy. The memory required for union is decided by the longest field in it.
The syntax is as follows
Union <identifier>
{
datatype1 element1;
datatype2 element2;
};

Example:
union example2
{
 int a; // 4 bytes
 long b; // 8 bytes
 char c; // 1 byte
 double d; // 8 bytes
};
The memory that is allocated for example2 union is 8 bytes which is the longest required memory space.

18. Write a C program to find out a solution for given quadratic equation.

Program:

```
#include<stdio.h>
#include<math.h>
int main()

{
float a,b,c,r,r1,r2,d;
printf("\nEnter the co-efficients\n");
scanf("%f%f%f",&a,&b,&c);
d=(b*b)-(4*a*c);
printf("\nThe value of d is  %f",d);
if(d<0)
printf("\nThe root are not real");
if(d>0)
```

```
{r1=(-b+sqrt(d))/(2*a);
r2=(-b-sqrt(d))/(2*a);
printf("\nThe root one is    %f",r1);
printf("\nThe root two is    %f",r2);
}
if(d==0)
{
r=-b/(2*a);
printf("\nThe roots are equal\n");
printf("Root1=Root2=%f\n",r);
}
return 0;
}
```

Output:

```
Enter the coefficients:
2 3 4
The value of d is -23.0000
The roots are not real
```