USN | | | | | | | | | | |

Internal Assessment Test 2 – March 2023

Sub:	Principles of Programming using C					Sub Code:	22POP13	Branch:	CSE	
Date:	06/02/2023	Duration:	90 mins	Max Marks:	50	Sem / Sec:	I / [All sections]			OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	Explain the different types of loops in 'C' with syntax and example. Loops in C programming language are a conditional concept used to execute a line or block of code consecutively. In C programming, there are three loops: For Loop, While Loop, and Do While Loop. Loops in C can also be combined with other control statements such as the Break statement, Goto statement, and Control statement. These loops can be used anywhere in the program, in either entry control or exit control units. **Different Types of Loops** There are 3 different types of Loops in C • **While Loop** • **Do While Loop** • **For Loop** **1. While Loop** In this, the condition is evaluated before processing the loop's body. Only the loop's body is executed if the condition is true. Then the control goes back to the beginning after completing the loop once. The statements in the loop will be executed again, and if the condition is true and checked, this process goes on until the condition becomes false. The control will go out of the loop if the condition is false. After completion of the loop, the control will go to the statement immediately after the loop, and the body can contain more than one statement. The	10	CO2	L2

curly braces are not that important if it has only one statement. If the condition is not true in the while loop, then loop statements won't get executed.

Syntax:

```
while (condition) {
statements;
}
```

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int num=1;
while(num<=5)
{
printf("%d\n",num);
num++;
}
return 0;
}
```

Output:

It will print the numbers from 1 to 5 like below.

2. Do While Loop

In this loop, the statements the loop need to be executed at least once. After that, it checks the condition. If the condition is true, it will again

have executed the loop; otherwise, it will exit it. It is known as an exit-controlled loop. It is similar to a while loop, and the condition is always executed after the body of the loop. The while loop is performed only when the condition is true, but sometimes the statement must be conducted at least once, so the **do-while loop** has to be used. The difference between while and do-while loop is that in the while loop, while is written in the beginning, and do-while, the condition is mentioned at the end and ends with a semicolon (;).

Syntax:

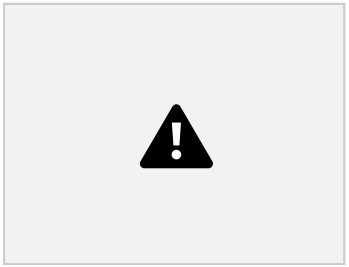
```
do {
statements
} while (expression);
```

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int num=1;
do
{
printf ("%d\n",2*num);
num++;
}
while(num<=5);
return 0;
}
```

Output:

The output of the above program is:



3. For Loop

It executes the set of statements until the time a particular condition is accomplished. It is known as the Open-ended loop. In For loop, we can have more than one initialization or increment/decrement, separated using a comma operator and one condition. For loop is used to evaluate the initialization part first, checking the condition for true or false. If the condition is true, it executes the statements of for loop. After that, it evaluates the increment or decrement condition until the condition becomes false it repeats the same steps. It will exit the loop when the condition is false.

Syntax:

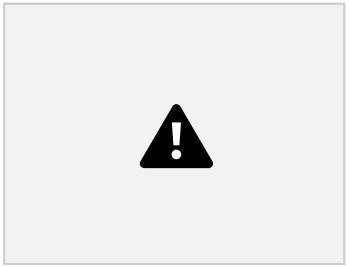
```
for (initial value; condition; incrementation or
decrementation )
{
statements;
}
```

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int number;
for(number=1;number<=5;number++)
{
printf("%d\n",number);
}
return 0;
```

```
}
```

Output:



There are nested For loops in which there is the outer For loop and inner loop. In **this nested loop**, the inner loop is repeated for the times for a given condition of outer loop iteration.

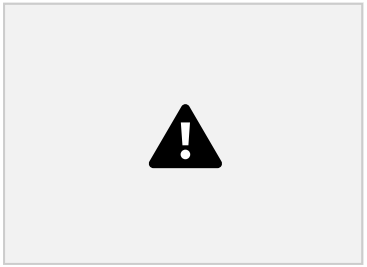
```
for(initialization;                    condition;
increment/decrement)
{
for(initialization;                    condition;
increment/decrement)
{
statement ;
}
}
```

Example:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int i, j;
for(i = 1; i < 5; i++)
{
printf("\n");
for(j = i; j > 0; j--)
{
printf("%d", j);
}
}
```

```
}
```

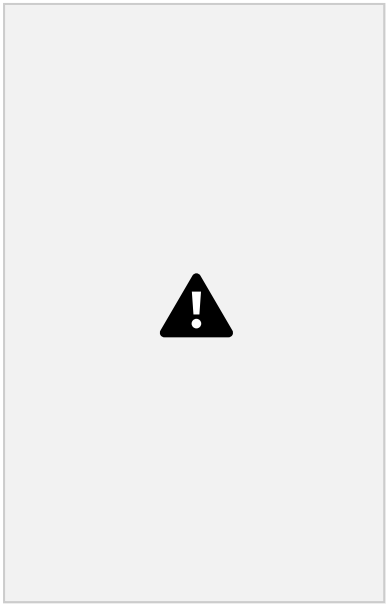
Output:



Other Example:

```
#include <stdio.h>
#include<conio.h>
int main() {
int i, j;
int table = 2;
int max = 5;
for (i = 1; i <= table; i++) {
for (j = 0; j <= max; j++) {
printf("%d x %d = %d\n", i, j, i*j);
}
printf("\n");
}}
```

Output:



Control Statements

Some loop control statements **need** to be **used in loops** for different purposes and to achieve the end result. Below are the different statements that are used:

Break statement

The break statement is used to exit the loop immediately after executing a particular statement for a specific condition.

Syntax:

```
While (Condition)
{ Statement 1; statement 2;
If (Condition)
{ break;}
Statement 3; }
```

Continue Statement

It generally skips the statements according to the condition. It is used to send the control directly to the condition and to continue the loop process. For a particular condition, it skips the current loop or statements and enters into a new loop or condition.

Syntax:

```
While (Condition)
{ Statement 1; statement 2;
If (Condition)
{ continue;}
Statement 3; }
```

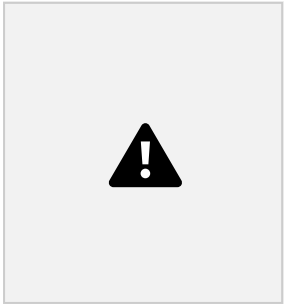
Goto statement

It is used to transfer the protocol to a labeled statement.

Example:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int number;
number=0;
repeat:
printf ("%d\n",number);
number++;
if(number<=5)
goto repeat;
return 0;
}
```

Output:



2	Explain the switch statement with syntax. Write a program for simple calculator using switch and explain the same.
	
	#include <stdio.h>
	
	int main() {
	
	char op;
	double first, second;
	printf("Enter an operator (+, -, *, /): ");
	scanf("%c", &op);
	printf("Enter two operands: ");
	scanf("%lf %lf", &first, &second);
	
	switch (op) {
	case '+':
	printf("%.1lf + %.1lf = %.1lf", first, second, first + second);
	break;
	case '-':
	printf("%.1lf - %.1lf = %.1lf", first, second, first - second);
	break;
	case '*':
	printf("%.1lf * %.1lf = %.1lf", first, second, first * second);
	break;
	case '/':
	printf("%.1lf / %.1lf = %.1lf", first, second, first / second);
	break;
	// operator doesn't match any case constant
	default:
	printf("Error! operator is not correct");
	}
	
	return 0;
	}
		10	CO 2	L2

3	Write a 'C' program to find roots of quadratic equation. # include<stdio.h> # include<conio.h> # include<math.h>	10	CO 2	L3

```
main (){
  float a,b,c,r1,r2,d;
  printf (“enter the values of a b c”);
  scanf (“ %f %f %f”, &a, &b, &c);
  d= b*b – 4*a*c;
  if (d>0){
    r1 = -b+sqrt (d) / (2*a);
    r2 = -b-sqrt (d) / (2*a);
    printf (“The real roots = %f %f”, r1, r2);
  }
  else if (d= =0){
    r1 = -b/(2*a);
    r2 = -b/(2*a);
    printf (“roots are equal =%f %f”, r1, r2);
  }
  else
    printf(“Roots are imaginary”);
  getch ();
}
```

4 What is an array? Explain the declaration and initialization of one and two dimensional arrays with example. 10 CO 3 L2

dimensional array

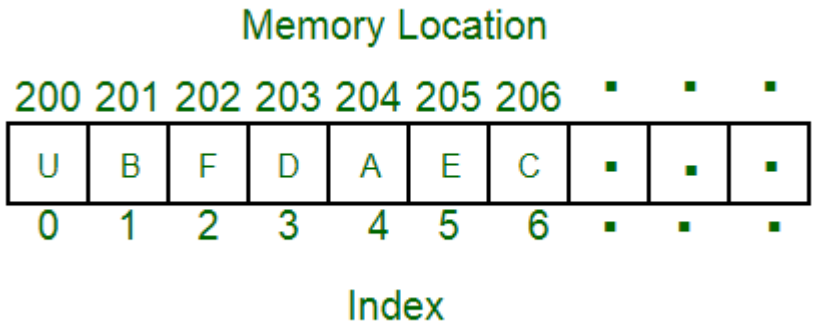
Array is a data structure that is used to store variables that are of similar data types at contiguous locations. The main advantage of the array is random access and cache friendliness. There are mainly three types of the array:

- One Dimensional (1D) Array
- Two Dimension (2D) Array
- Multidimensional Array

One Dimensional Array:

- It is a list of the variable of similar data types.
- It allows random access and all the elements can be accessed with the help of their index.
- The size of the array is fixed.
- For a dynamically sized array, vector can be used in C++.

- Representation of 1D array:



Two Dimensional Array:

- It is a list of lists of the variable of the same data type.
- It also allows random access and all the elements can be accessed with the help of their index.
- It can also be seen as a collection of 1D arrays. It is also known as the Matrix.
- Its dimension can be increased from 2 to 3 and 4 so on.
- They all are referred to as a multi-dimension array.
- The most common multidimensional array is a 2D array.
- Representation of 2 D array:

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Difference Table:

Basis	One Dimension Array	Two Dimension Array
Definition	Store a single list of the element of a similar data type.	Store a 'list of lists' of the element of a similar data type.
Representation	Represent multiple data items as a list.	Represent multiple data items as a table consisting of rows and columns.
Declaration	The declaration varies for different programming language: 1. For C++, ***datatype variable_name[row]*** 2. For Java, ***datatype [] variable_name= new datatype[row]***	The declaration varies for different programming language: 1. For C++, ***datatype variable_name[row][column]*** 2. For Java, ***datatype [][] variable_name= new datatype[row][column]***

	Dimension	One	Two		
	Size(bytes)	size of(datatype of the variable of the array) * size of the array	size of(datatype of the variable of the array)* the number of rows* the number of columns.		
	Address calculation.	Address of a[index] is equal to (base Address+ Size of each element of array * index).	Address of a[i][j] can be calculated in two ways row-major and column-major 1. **Column Major:** Base Address + Size of each element (number of rows(j-lower bound of the column)+(i-lower bound of the rows)) 2. **Row Major:** Base Address + Size of each element (number of columns(i-lower bound of the row)+(j-lower bound of the column))		

	Example
	
	int arr[5]; //an array with one row and five columns will be created.
	
	{a , b , c , d , e}
	
	int arr[2][5]; //an array with two rows and five columns will be created.
	
	a b c d e
	
	f g h i j
	
	<u>Applications of Arrays</u>:
	
	● 2D Arrays are used to implement matrices.
	● Arrays can be used to implement various data structures like a heap, stack, queue, etc.
	● They allow random access.
	● They are cache-friendly.			

5	Explain the use of break and continue statement in loops with example.
	The one-token statements `continue` and `break` may be used within loops to alter control flow; `continue` causes the next iteration of the loop to run immediately, whereas `break` terminates the loop and causes execution to resume after the loop. Both control structures must appear in loops. Both `break` and `continue` scope to the most deeply nested loop, but pass through non-loop statements.
	
	Although these control statements may seem undesirable because of their goto-like behavior, their judicious use can greatly improve readability by reducing the level of nesting or eliminating bookkeeping inside loops.
	
	Break Statements
	
	When a `break` statement is executed, the most deeply nested loop currently being executed is ended and execution picks up with the next statement after the loop. For example, consider the following program:
	
	`while (1) {`
	`  if (n < 0) break;`
	`  foo(n);`
	`  n = n - 1;`
		5	CO 2	L2

	}
	
	The `while~(1)` loop is a "forever" loop, because `1` is the true value, so the test always succeeds. Within the loop, if the value of `n` is less than 0, the loop terminates, otherwise it executes `foo(n)` and then decrements `n`. The statement above does exactly the same thing as
	
	`while (n >= 0) {`
	`  foo(n);`
	`  n = n - 1;`
	`}`
	
	This case is simply illustrative of the behavior; it is not a case where a `break` simplifies the loop.
	
	Continue Statements
	
	The `continue` statement ends the current operation of the loop and returns to the condition at the top of the loop. Such loops are typically used to exclude some values from calculations. For example, we could use the following loop to sum the positive values in the array `x`,
	
	`real sum;`
	`sum = 0;`
	`for (n in 1:size(x)) {`
	`  if (x[n] <= 0) continue;`
	`  sum += x[n];`
	`}`			

	Explain the syntax of nested 'if' statements. Write a 'C' Program to find largest of three numbers using nested 'if' statement.	5	CO 2	L3
6	Write a 'C' Program to find the sum of odd numbers 'n' natural numbers using do 'while' loop.	5	CO 2	L3
	Write a 'C' program to search a key element using linear search.	5	CO 3	L3

CI CCI

HOD