Assignment-2

1 Explain the syntax of multiple if... else statement.

Multiple if...else is a decision making statement in programming laugvage.

This is decide the direction of the flow of program exeution in programming.

In multiple if else statement is present inside the body of program onther if or else statement.

Multiple If in c programming is helpful to if you want to check the condition inside a condition.

In sitution arise in programming also where we need to make some decisions and based on these decisions.

Syntax :

```
if (Condition - 1)
{
  if (Condition - 2);
  {
    statement - 1;
  }
  else
  {
    statement - 2;
  }
}
else
  {
    statement - 3;
  }
```

-> If the condition - 1 is False, than the statement 3 will be executed.

-> If the condition - 1 is true, than it continues to perform the second condition.

Second Condition is true, than the statement 1 is executed and False, than the statement 2 is executed.

2 Explain the swith statement with example.

Switch statement is tests the value of a variable and compares it with multiple case.

Switch case is used to perform different actions based on different conditions.

Switch is allow a value to change control of execution.

The switch statement is a multiway branch statement.

=> In switch case there are some important keywords.

1 Break Statement:

This keyword is used to stop the execution inside a switch cases.

2 Default Statement:

Specify the set of ~~se~~ statement to

execute if there is no case match.

Example: Calculate Grade of given marks.

```
#include <stdio.h>
void main()
{ int grade;
printf("Enter Marks");
scanf("%d", &grade);
switch(grade)
{
 case 10;
 case 9;
 case 8;
 printf("A grade");
 break;

 Case 7;
 Case 6;
 printf("B grade")
 break;

 default: printf("Fail");
 }
 getch();
}
```

3 Explain operator precedence and associativity.

=> Operator Precedence

In Precedence program decides how an expression is evaluate.

Precedence ~~opet~~ operators with the higest precedence appear at the top of the table.

Some time many expressions are together than we decides how will be evaluated first.

Than higher Precedence operators will be evaluated first and lower Precedence will be evaluated last.

Ex. 10 + 20 * 30

lower Precedence

higher Precedence

That's why multiplication operator evaluate first and addition operator evaluate last.

=) Associativity:

Associativity is used when two operators of same precedence apper in an expression.

Associativity can be either left to Right or Right to left.

Precedence associativity is left to Right.

It only used when there are two or more operators of same precedence.

Ex. 100 / 10 * 10

Here '/' and '*' both have same Precedence.

In Precedence associativity is left to Right. That's why division operator operat first.

4 Explain use of getchar() and putchar() Functions.

=> getchar() Function

Getchar Function is a part of the <stdio.h> header file.

getchar is a non standard function.

It accept a single input from the user.

getchar function is read a single character.

It Declared by int getchar(void)

=> putchar() Function:

Putchar() is a output Function.

Putchar function is used to write one character

putchar() function accepts one argument of character type.

It Declared by int putchar(int char).

5 Difference between type casting and type conversation.

	Type Casting	Type Conversation
1	It converted into anthor data type by a programmer.	It converted into anthor data type by a complier.
2	Type Casting is incom-patible.	Type Converted is Compatiable.
3	It needed casting operator.	It is no needed casting operator.
4.	Type Casting is narrowing conversion.	Type Conversation is widening conversion.

	Type Casting	Type Conversation.
5	Type casting data type is smaller than source data type.	Type conversation data type is no smaller than source data type.
6	Type Casting is more use in coding and competitive programming.	Type Conversation is less use in coding and competitive programming.
7	It is more efficient.	It is less efficient
8	It is more reliable.	It is less reliable.

6 What is loop and explain different types of looping structures.

A Loop is a sequence of instructions in programming.

loop is use for when we need to repeatedly exeute a block of statemant.

In loop, the statemant needs to be written once.

loop instructions is repeated untill a certain condition is reached.

In loops allows to use a group statemant for multiple times.

loop use to exeute the block statemant of code serveal times.

=> There are two types of looping structures.

1 Entry Controlled loops
2 Exit Controlled loops

1 Entry Controlled loop:

In entry controlled loop test of condition is tested before entering the loop.

Ex. For loop and while loop

2 Exit Controlled loop:

In Exit controlled loops test of condition is tested at the end of loop body.

Therefore, the loop body will execute atleast once, whether the condition is true or false.

Ex. do while loop.

7 Write a syntax of for loop. How it differs from while loop?

For loop is entry controlled loop.

For loop is the test condition is tested before entering the loop body.

=) Syntax:

```
for(Initialization ; test ; Update
    Expressions    expre.  Expressions)
{
   loop & body
   statement
}
```

-> Initialization: We have to initialize the loop counter to some value

Ex: int i = 1;

-> Test Expression: We have to test the condition. If condition is true then we will execute the loop body. and go to update Expression.

Ex. i <= n;

-> Update Expression: After executing loop counter value is increment or decrement by some value.

Ex. i++;

=> While loop:

While loop Initialization is always outside the loop.

Increment can be done before or after the execution of the statement.

In while loop condition may be expression or non zero value.

While loop is used for complex initialization.

Syntax: while(condition)
{
statement ;
}

10 Difference between while and do while loop.

	While Loop	Do While Loop
1	Condition is checked first then statement executed.	Codition is checked after the executed.
2	While loop is may or may not executed atlest once.	do while loop is executed atlest once.
3	No Semicolon at the end of while.	Semicolon at the end of do while.
4	Bracketes are not required.	Bracketes are required.
5	Entry Controlled loop	Exit Controlled loop.

6	Syntax: while(condition)
	{
	statement;
	}
		Syntax: do
		{ statement;
		}
		while(condition);

11. Difference between Entry Controlled and Exit Controlled loop.

	Entry Controlled loop	Exit Controlled loop
1	Check condition first then executes.	Check condition after the executes
2	The body of loop may or may not executed.	The body of loop atlest once executed.
3	Ex. For and while loop	Ex. do while loop.
4	If test condition is False then no executed	If test condition is False then executed.
5	No Semicolon after for and while loop.	Semicolon after do while loop.

SMVS
Page No.
Date: / /

8 Explain Break statement and continue statement with example.

=> Break Statement:

Break statement is used to exit from the loop in program.

Break statement is used in switch case and loops.

Syntax: Break;

Break statement is control over the program execution.

When Break statement is applied then execution of loop is stop.

Ex.

```
#include <stdio.h>
  void main()
{ int i = 0;
  while (1)
  { printf("%d", i);
```

```
    i++;
      if(i==3)
      break;
    }
    printf("Number");
}
```

=> Continue statement:

Continue Statement is not used to exit from the loop.

Continue statement is used only in loops.

Syntax: Continue;

When Continue statement is applied than statement after it are skipped.

Ex.

```
#include<stdio.h>
int main()
{ int i, j;
  for(j=0; j<=5; j++)
  {
    if(j==3)
```

```
{ continue;
  }
  printf("%d", j);
 }
 return 0;
}
```

9 Compare break statement and Continue statement.

	Break Statement	Continue Statement.
1	It is used to exit from loop.	It is used not to exit from loop.
2	It is use in switch and loops.	It is use only in loops.
3	It control program execution.	It is not control program execution.
4	denoted : break;	denoted : continue;
5	When it applied then loop is close.	when it applied then statement is skipped.

12. Write a program to Find power of a number using For loop.

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

 printf("Enter the n \n");
 scanf("%d", &n);

 printf("Enter the Exponent \n);
 scanf("%d" &Exponent);

 for(i=1; i<=Exponent; i++)
 {
   power = Power * n;
 }
 printf("The Final value of %d Power
          %d = %ld", n, Exponent, Power);

 getch();
 return 0;

}
```

Output :

Enter the n 2
Enter the Exponent 2
The Final value of 2 Power 2 = 4

13 Write a program to print all natural numbers from 1 to n - using while loop.

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

{ int i, n;

  printf("All natural numbers from 1 to: ");
  scanf("%d", &n);

  i = 1;
  while(i<=n)
  {
    printf("%d\t", i);
    i++;
  }
  return 0;

}
```

Output:

All natural numbers from 1 to 10

1 2 3 4 5 6 7 8 9 10

14 Write a program to print all natural numbers in reverse using while loop.

```
#include <stdio.h>
#include <conio.h>
int main()
{
int n, i;

printf("Enter the last value ");
scanf("%d", &n);

i = n;

while(i >= 1)
{
printf("%d \t", i);
}
return 0;
}
```

Output:

Enter the last value 4

4 3 2 1

15 Write a program to print all alphabets from a to z using while loop.

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

{ char ch='a';
  clrscr();

  printf("Alphabets from a-z:\n");
  while(ch<='z')
  {
    printf("%c\t",ch);
  }
  getch();

}
```

output:
Alphabet from a to z:
a b c d e f g h i j k l m

n o p q r s t u v w x y z

16 Write program to print all even number between 1 to 100 using while loop.

```
#include <stdio.h>
#include <conio.h>
int main()
{ int i;

  printf("Even number between 1 to 100\n");

   i = 2;
   while (i <= 100)
   {
     printf("%d", i);
   }
      i = i + 2;
   }
   return 0;
}
```

Out put:

Even number between 1 to 100:

2 4 6 8 10 12 14 16 18 20
22 24 26 28 30 32 34 36 38 40
42 44 46 48 50 52 54 56 58 60
62 64 66 68 70 72 74 76 78 80
82 84 86 88 90 92 94 96 98 100

17 Write program to print all add number between 1 to 100 using while loop.

```
#include <stdio.h>
#include <conio.h>
int main ()
{ int i=1;

  printf (" Odd number between 1 to 100\n");

  while (i<=100)
  {
   printf (" %d \t", i);
    i = i + 2;
  }
  return 0;
}
```

Output:

Odd number between 1 to 100

1 3 5 7 9 11 13 15 17 19
21 23 25 27 29 31 33 35 37 39
51 53 55 57 59 61 63 65 67 69
71 73 75 77 79 81 83 85 87 89
91 93 95 97 99

18 Write a program to print the Fibonacci series (1, 1, 2, 3, 5, 8, 13, ---)

```
#include <stdio.h>
#include <conio.h>
int main ()
{ int t1=0, t2=1, i=0, n;
  printf ("Enter the number: \n");
  scanf ("%d", &n);
  printf (" Fibonacci Series: ", t1, t2);
    i = t1 + t2;
```

```
while(i<=n)
{printf("%d\t",i);
    t1=t2;
    t2=i;
    i=t1+t2;
}
 return 0;
}
```

Out put:

Enter the number : 20

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13

20 What is the purpose of scanf() and printf() function.

=) scanf()

scanf function get the input from user.

It reads input for number and other datatypes from standard input.

scanf function stores a series of character.

scanf function declared in c library.

=> printf()

Printf function get the output ~~for~~ from display.

printf function declared in c library.

printf function prints the character on standard output.