Assignment - II

Name - Pooja Langaakhandi
Div - A Sem - 1st
Roll No - 015BL23S0047

1] What is Loop ?

→ A loop statement allows us to execute a statement or group of statements multiple times.

2] Name the three key steps in looping techniques.

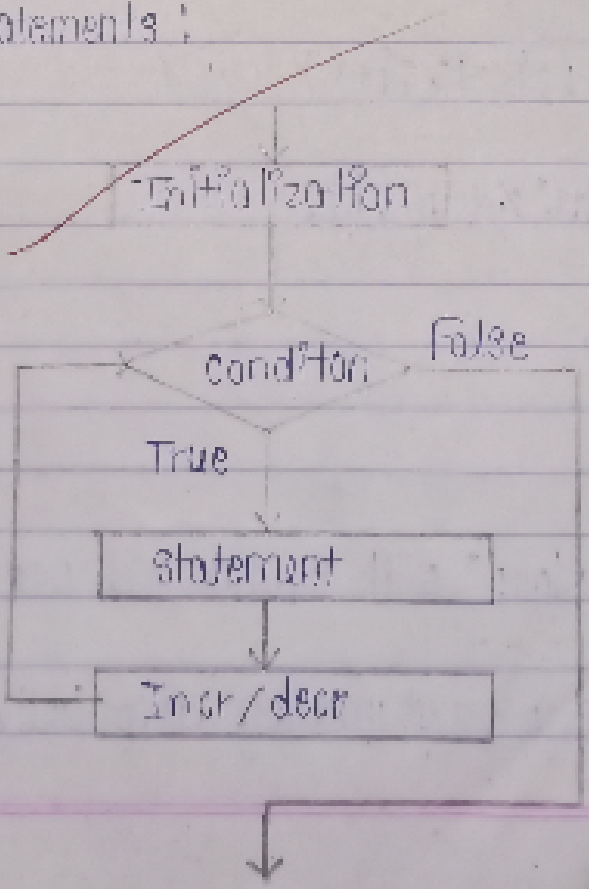
→ The three key steps in looping techniques are :-
i] Initialization
ii] Condition
iii] Increment

3] Explain "for loop" with Syntax and example.

→ For loop :-

- A for loop is a repetition control structure that helps us to (or allows) the statements or a part of the program specified number of times.
- Syntax :-

```
for (initialization ; condition ; increment)
{
    statements ;
}
```



- The flow of control in a 'for' loop :
  - The initialization step is executed first, and only once. This step allows you to declare and initialize any loop control variables.
  - Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
  - After the body of the 'for' loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables.

- Example :-

```
// C program to print 1 to 10 numbers
#include<stdio.h>
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
      printf("%d\n",i);
    }
  return 0;
}
```

4] Explain "while loop" with syntax and example

→ While loop :-

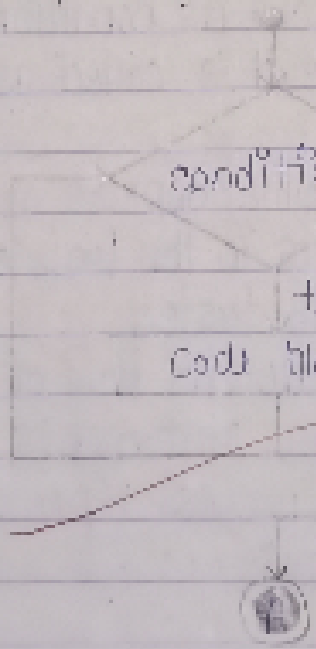
- A while loop in programming repeatedly executes a

target statement as long as a given condition is true.
- It is entry controlled

Syntax :-

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

- Here, statements may be a single statement or a block of statements.
- The condition may be any expression, and true is any non-zero value.
- The loop iterates while the condition is true. The condition is checked at the entry of the loop and since it is called as the entry controlled loop.
- When the condition becomes false, the program control passes to the line immediately after the loop.



- Example :-

// C program to print 1 to 10 natural numbers using

while.

```
#include<stdio.h>
int main()
{
  int a=1;
  while(a<=10);
  {
     printf("%d\t", a);
     a++;
  }
  return 0;
}
```

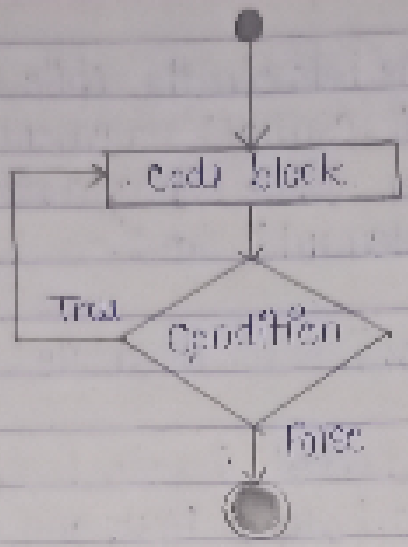
5) Explain "Do while loop" with syntax and example ?

→ Do While loop :-

- The do while loop checks its condition at the bottom of the loop and hence it is called as an exit Controlled loop.
- The conditional expression appears at the end of the loop, so the statements in the loop executes once before the condition is tested.
- If the condition is true, the flow of control jumps back up to do, and the statements in the loop executes again. This process repeats until the given condition becomes false.

Syntax :-

```
do
{
   Statements;
}while(condition);
```



• Example :-

```
// C program to print numbers from 10 to <20
#include<stdio.h>
int main()
{
    int a=10;
    do
    {
        printf("Value of a : %d\n", a);
        a=a+1;
    }while(a<20);
    return 0;
}
```

Output :-

```
Value of a : 10
Value of a : 11
Value of a : 12
Value of a : 13
Value of a : 14
Value of a : 15
Value of a : 16
Value of a : 17
Value of a : 18
Value of a : 19
```

6) State the difference between the while (entry control) and do-while (exit control) statement?

→ Difference between while (entry control) and do-while (exit control) statements are :-

while (entry control)	do-while (exit control)
1) Test condition is checked first, and then loop body will be executed.	1) Loop body will be executed first, and then condition is checked
2) If Test condition is false, loop body will not be executed	2) If test condition is false, loop body will be executed once.
3) for loop and while loop are the example of Entry Controlled loop	3) do while loop is the example of exit control an loop.
4) Entry controlled loops are used when checking of test condition is mandatory before executing loop body.	4) Exit controlled loop is used when checking of test condition is mandatory after executing the loop body.
5) It is an entry controlled loop	5) It is an exit controlled loop
6) In case of a single statement, one do need to add brackets.	6) Brackets are always needed.

Q) Explain jump in loops with syntax and example. (break, continue & goto statements).*

→ 1) Break statement :-

- Syntax :- 'break'
- Example :-

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

output : 0 1 2

- Explanation :-

The 'break' statement is used to exit a loop prematurely when a specific condition is met. In the example, the loop stops when 'i' becomes 3.

2) Continue statement :-

- Syntax :- 'continue'
- Example :-

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

output :- 0 1 3 4

- Explanation :-

The 'continue' statement skips the rest of the loop's code and moves to the next iteration. In this case, it skips the print statement when 'i' is 2.

3) Goto statements :-

- Syntax :- goto label;
- Example :- ....

Table : Statement :

• Example.

```
for (int i=0; i<5; i++)
{
  if (i == 3)
  {
    goto end_loop;
  }
  printf("%d", i);
}
```

• Explanation :-

The 'goto' statement transfers control to a 'labeled' statement ('endloop' in this case). Quite it can be used, it's often considered bad practice due to the potential for creating spaghetti code.

8] What is nesting of loop ? Give example.

→ Nested loop :- Nesting of loops also known as looping of statement inside another loop.

Example :-

// C program to print multiplication table of n natural numbers

```
#include<stdio.h>
int main()
{
  int n;
  printf("Enter the value of n:");
  scanf("%d", &n);
  for(int i=1; i<=n; i++)
```

```
    {
      for(int j=1 ; j<=10; j++)
      {
        printf("%d\t", (i*j));
      }
      printf("\n");
    }
    return 0;
}
```

3] What is an array ?

→ "An array is a collection of same types of data (homogeneous) items that are stored in the contiguous memory locations in computer memory under a common variable name."

12] Explain the classification of an array.

→ The classification of an array are :-

i) One dimensional array
ii) Two dimensional array
iii) Multi dimensional array

i] One dimensional array :-

- A dimensional array is an array with n elements where each element is stored in consecutive memory location. One dimensional array has a single subscript or index.
- Declaration of one dimensional array

Syntax :-

datatype variable_Name[array_size];

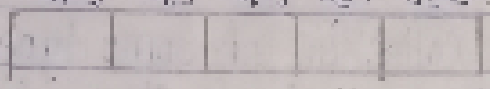
datatype :- defines the types of data items in the array.

variable :- defines the unique name of an array.

arraySize :- defines the number of elements contained in the array.

Example :- int a[5];

5 consecutive memory locations are reserved by a common name a.



ii) Two dimensional array :-

A two dimensional array is an array with two subscripts. The elements here are stored in rows and column format. The total number of elements in a two dimensional array will be number of rows x columns.

Declaration of two-dimensional array

Syntax :-

datatype variable_Name[rows][columns];

datatype :- defines the types of data items that the array will be holding

variable / name :- defines the unique name of an array.

rows :- define the number of Rows

Columns :- define the number of columns

Example :-
int a[3][4];

c) Multi dimensional array :-
A multi dimensional array is an array with more than two subscripts or dimension.

Example :-
int a[3][2][2];

d) State the difference between an array and a variable.

=>

	ARRAY	VARIABLE
•	Array holds multiple values	• Variable hold a single value.
•	An array is the set of no of variable denoted by a the unique name.	• Variable space defined by its function (char, int, float etc) in which you can store some value.
•	The elements of the array are treated as individual entities	• The variable is simple scalar variable such as as int
•	Syntax :- datatype variable_name - [array_size];	• Syntax :- datatype variable_list;

• Example :- int num[20];	• Example :- float x, y, z; int a;

[12] Give the syntax for declaring one dimensional array. Give example.

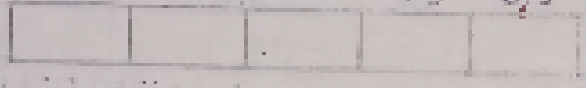
→ The syntax for declaring one dimensional array :-

• Syntax :-

datatype variable_name [array_size];

• Example :-

int a[5];



[13] Give the syntax for declaring two dimensional array. Give example.

→ Declaration of two-dimensional array :-

• Syntax :-

datatype variable_name [rows][columns];

• Example :-

int a[3][4];

Six twelve consecutive memory location are reserved by a common name a,

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

(a) Mention the various methods of one dimensional array with initialization with example.

→ Initialization of one dimensional array :-

Once the array is declared it must be initialized otherwise it will contain a garbage value.

Syntax to initialize an array

datatype variable_name[size] = {list};

The values in the list are separated by commas

Example :-

int a[5] = {10, 20, 30, 40, 50};

This declaration will create an array like this :

a[0]	a[1]	a[2]	a[3]	a[4]
10	20	30	40	50

Example :-

To initialize character arrays

char a[5] = {'a', 'b', 'c', 'd'};

This declaration will create an array like this :

a[0]	a[1]	a[2]	a[3]	a[4]
a	b	c	d	

In the above example :- we have declared an array of five (5) characters but initialized it with only four(4). Hence the last element of the array will be left blank.

15] Explain the various methods of two-dimensional array with initialization with example.

→ Initialization of two-dimensional array :

Example :-

```
int a[3][3] = {
                {10, 20, 30},
                {40, 50, 60},
                {70, 80, 90},
              };
```

or a 2 dimensional array can be initialized as the following :

```
int a[3][3] = {{10, 20, 30}, {40, 50, 60}, {70, 80, 90}};
```

This declaration will create an array like this :

	Column (0)	Column (1)	Column (2)
Row (0)	10	20	30
Row (1)	40	50	60
Row (2)	70	80	90

16] What is a string ? Give example.

→ A string is a sequence of characters which is treated as a single data type in C. It is an array of characters.

Example :-

```
char str[5] = "hello";
```

(1) Explain the different methods to read and write strings. (Character array).

→ String I/O function in C Language. The following are the input and output functions of string in C:

Input functions :-
① scanf()
② gets()
③ getchar()

Output functions :-
① printf()
② puts()
③ putchar()

- The scanf() and printf() are generic I/O functions that they support all built-in data types. The scanf() accepts character by character from keyboard until either a new line ('\n') or blank space is found, which ever comes earlier.

- But gets() and puts() are specialized to scan and print only string data.

- The gets() accepts a string until a new line is found. That is it accepts white space & tab also.

Q] What is null character? State its use in a string.

→ The length of a string is always greater than the number of string characters by one because after a compiler assigns a character string to a character array, it automatically supplies a null character ('\0') at the end of the string.

Q] Explain the following string handling functions:

→

i] strcpy() :-

This function is used to copy the second string into the first string.

Syntax :-

```
strcpy(str1, str2);
```

ii] strcat() :-

This function is used to concatenate two strings.

Syntax :-

```
strcat(str1, str2);
```

Example :-

```
strcat("My", "program")
```

Output : myprogram

iii] strcmp() :-

This function is used to compare its first and second strings character wise

- It returns 0 if the strings are equal
- It returns a positive value if the first string is greater than second.

- It returns a negative value if the first string is less than second.

Syntax :-

```
Strcmp(str1, str2);
```

iv) Strlen() :-

This function is used to find the length of a string.

Syntax :-

```
Strlen(str);
```

Example :-

```
Strlen("program");
```

output : 8

v) Strrev() :-

This function is used to reverse the given string.

Syntax :-

```
Strrev(str);
```

Example :-

```
Strrev("Raj");
```

Output : jaR

Q9] Explain various string manipulation functions with syntax and example.

→ String manipulation functions are :

1) Strcat() :-

This functions is used to concatenate two strings.

Syntax :-

```
Strcat(str1, str2);
```

Example :-

strcat ("My", "program");

Output : Myprogram.

2] Strlen ():-

This functions is used to find the length of a string.

Syntax :-

Strlen ("program");

Output : 7

3] Strcpy () :-

This function is used to copy the second string into its first string

Syntax :-

Strcpy (Str1, Str2);

4] Strcmp () :-

this function is used to compare its first and second strings character wise.

- It returns 0 if the strings are equal.
- It returns a positive value if the first string is greater than second
- It returns a negative value if the first string is less than second

Syntax :-

Strcmp (Str1, Str2);

5] Strrev () :-

This function is used to reverse the given string.

Syntax :-

strrev(str);

Example :-

strrev("abc");

Output : "cba"

5] strlwr() :-

This function is used to convert the given string to lower case.

Syntax :-

strlwr(str);

Example :-

strlwr("PROGRAM");

output : program.

6] strupr() :-

This function is used to convert the given string to upper case.

Syntax :-

strupr(str);

Example :-

strupr("program");

Output : PROGRAM

20] Program to find factorial of a number.

==> // C program to find factorial of a number.

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

```
int n, i, fact=1;
printf("Enter an integer:");
scanf("%d", &n);
if(n<0)
    printf("Error! factorial of a negative
    number doesn't exist");
    printf("Invalid input enter a positive number
    \n");
else if(n==0)
    printf("Factorial of %d is %d \n", n, fact);
else
{
    for(i=1; i<=n; i++)
    {
        fact=fact*i;
    }
    printf("\n Factorial of %d is %d \n", n, fact);
}
```

22] Program to generate Fibonacci series

→ // C program to print Fibonacci series of n numbers using while.

```
#include<stdio.h>
int main()
{
    int i, n, t1=0, t2=1 nextTerm;
    printf("Enter the number of terms:");
    scanf("Fiben: %d", &n);
```

```
printf("Fibonacci series:");
for (i=1; i<=n; i++)
{
  printf("%d", t1);
  nextterm = t1 + t2;
  t1 = t2;
  t2 = nextterm;
}
return 0;
}
```

Q] Program to demonstrate string functions.

```
#include<stdio.h>
#include<string.h>
void main()
{
  char str1[20], str2[20], str3[20];
  printf("\n Enter first string:");
  gets(str1);
  printf("\n Enter second string:");
  gets(str2);

  printf("\n1. Length of first string : %d", strlen(str1));
  printf("\n2. Copy first string into third : %s", strcpy
        (str3, str1));
  printf("\n3. Compare first and second : %s",
      (strcmp(str1, str2)==0) ? "true" : "false");
  printf("\n4. Concatenate : %s \n", strcat(str1, str2));
}
```

2] Program to find length of string without using in-built function.

→
```
#include<stdio.h>
void main()
{
    char str[20];
    int i, length;
    printf("\n Enter the string :");
    gets(str);
    for(i=0; str[i] != '\0'; i++)
    {
        length = i+1;
    }
    printf("\n The length of the string is %d \n",
            length);
}
```

3] Program to reverse a string without using built in function.

⟹
```
#include<stdio.h>
void main()
{
    char str[25], rev[25]= "\0";
    int i, len = 0;
    printf("Enter the string \n");
    gets(str);
    for(i= 0; str[i] != '\0'; i++)
    {
        len++;
    }
```

```
    for(i=len-1; i>=0; i--)
    {
      rev[len-i-1] = str[i];
    }
    printf(" Reverse String is %s\n", rev);
}
```

2] Program to generate fibonacci series.

---> /* Program to generate Fibonacci Sequence.

```
#include<stdio.h>
void main()
{
    int t1=0, t2=1, i=2, nextterm, n;
    printf(" Enter a positive number: ");
    scanf("%d", &n);
    if(n==-1)
    {
      printf("\n Fibonacci Sequence: %d ", t1);
    }
    else
    {
      printf("\n Fibonacci Sequence: %d %d", t1, t2);
      while(i<n)
      {
          nextterm = t1+t2;
          scanf("%5d", nextterm);
          t1 = t2;
          t2 = nextterm;
          i++;
      }
    }
    printf("\n");
}
```