**LAB # 04**



**CSE102L Computer Programming Lab**

Submitted by: **Maaz Habib**

Registration No. :**20PWCSE1952**

Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**Engr. Abdullah Hamid**

Month Day, Year (May,30, 2021)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

To understand the programming knowledge using selection statements like “switch”

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case.

Syntax

The syntax for a **switch** statement in C++ is as follows:

switch(expression) {

case constant-expression :

statement(s);

break; //optional

case constant-expression :

statement(s);

break; //optional

// you can have any number of case statements.

default : //Optional

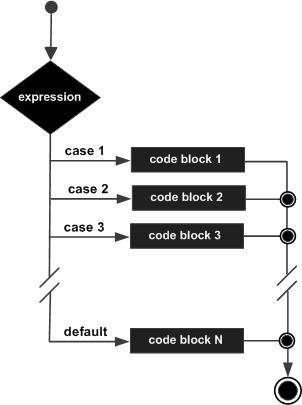
statement(s);

}

The following rules apply to a switch statement:

* The **expression** which are being used in a **switch** statement must have an integral type, or it should be of a class type in which the class has a single conversion function to an integral type.
* Any number can be used in case statements within a switch. Each case is followed by the value to be compared to and a colon.
* The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
* When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.
* When a break statement is reached, the switch terminates, and flow of control jumps to the next line following the switch statement.
* If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
* A **switch** statement can have an optional **default** case, which must appear at the end of the switch. No break is needed in the default case.

Flow Diagram:



To understand the programming using iteration statements like for Loop:

Syntax:

The syntax of a for loop in C++ is −

for ( init; condition; increment ) {

statement(s);

}

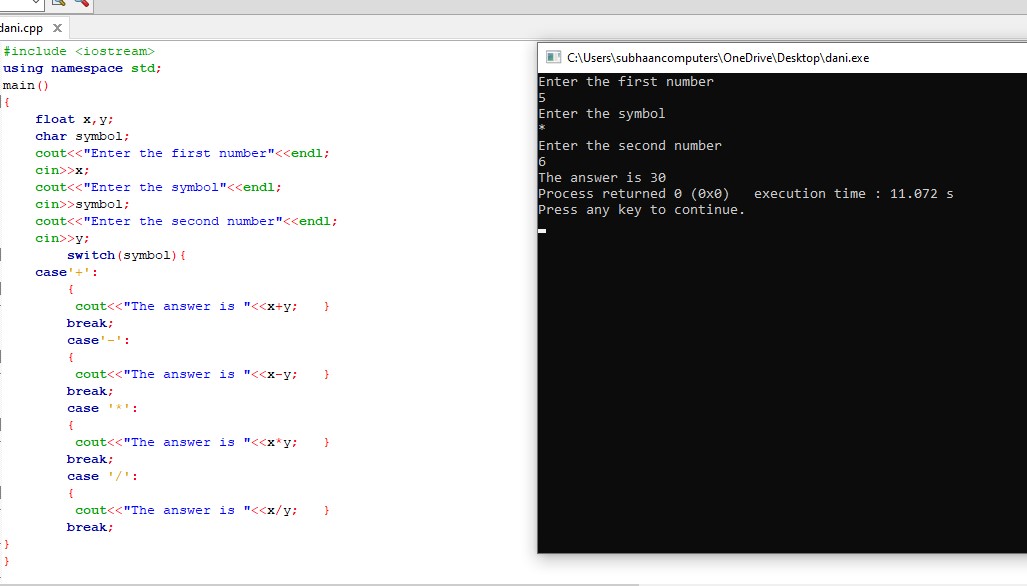
Here is the flow of control in a for loop:

* The **init** step is executed first, and only once. This step allows you to initialize any loop control variables. You shouldn’t put a statement here, until a semicolon appears.
* Next, the **condition** is evaluated. If the condition is true, then the body of the loop is executed. If the condition is false, then the body of the loop does not execute and 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.
* Increment statement can be left blank, as long as a semicolon appears after the condition.
* The condition is now evaluated again. If it is true, the loop executes and the process repeats itself. After the condition becomes false, the for loop terminates.

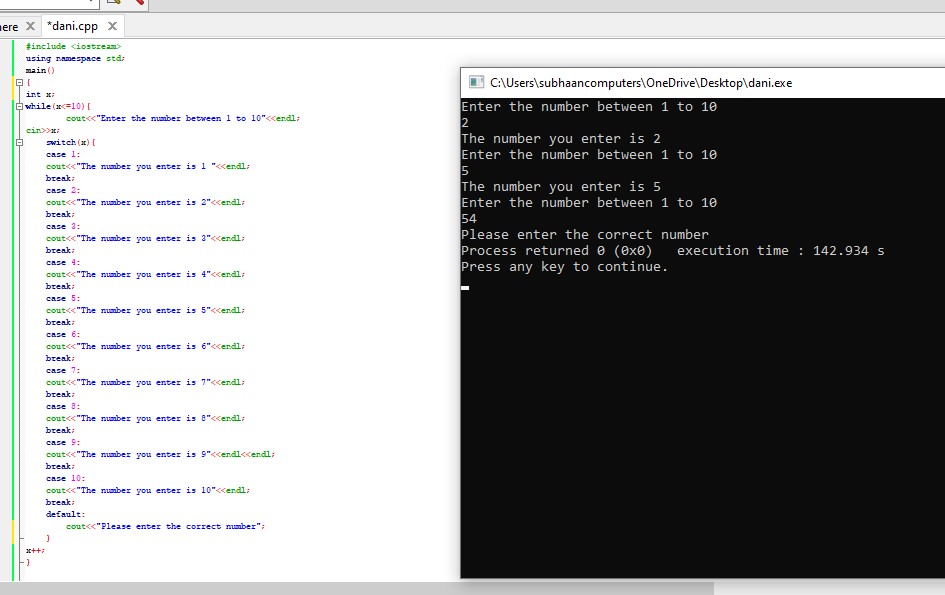
Flow Diagram:



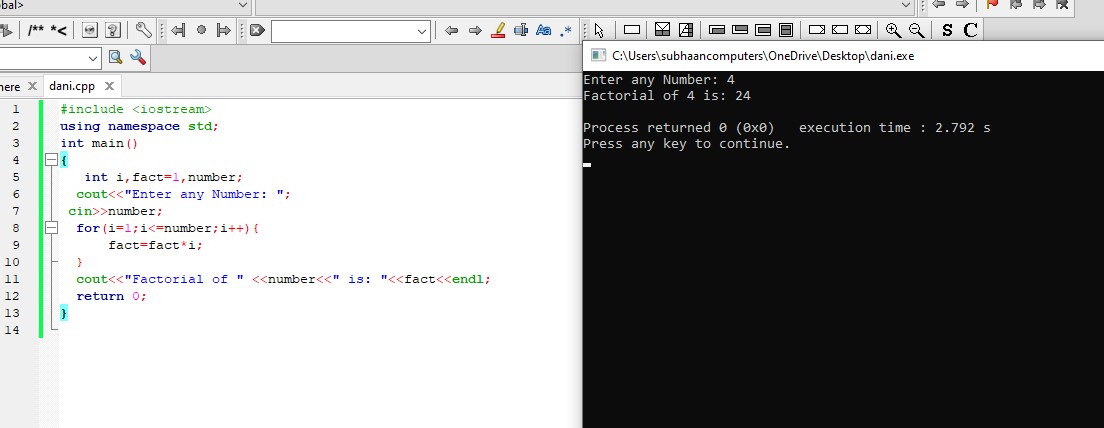
**Task No 01:**

****

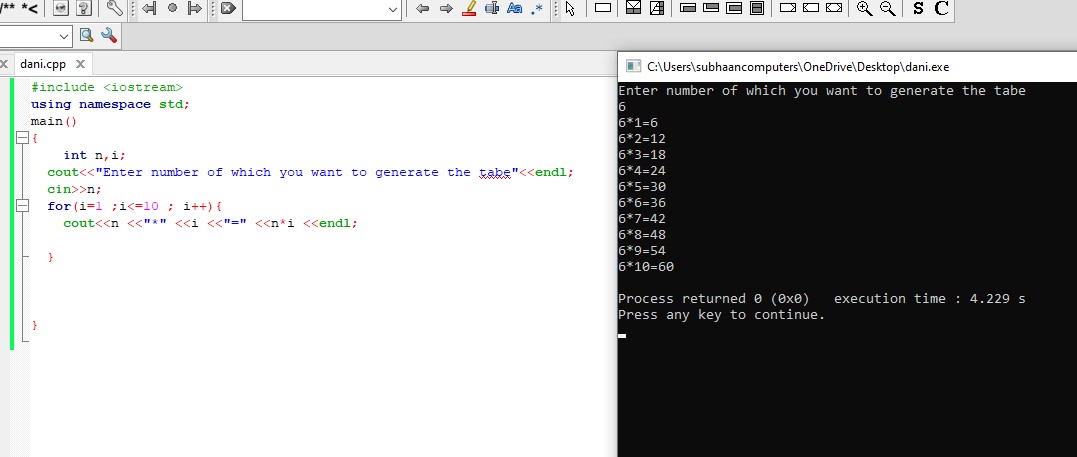
**Task No 02:**

****

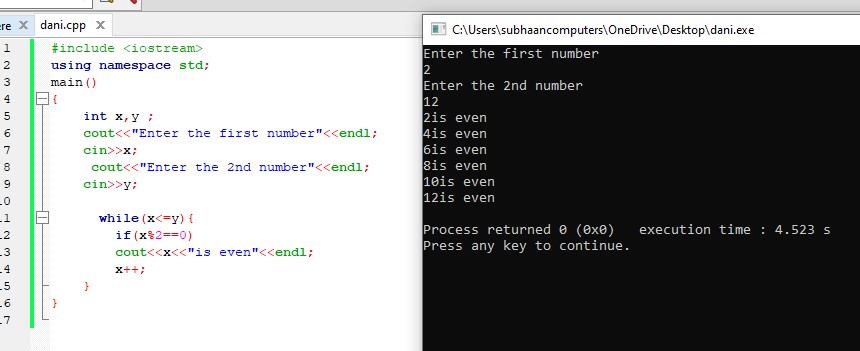
**Task No 03:**

****

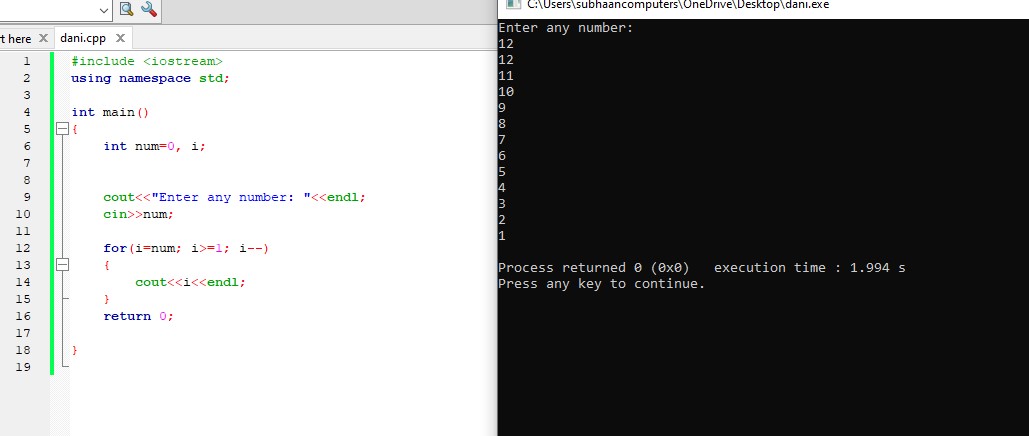
**Task No 04:**

****

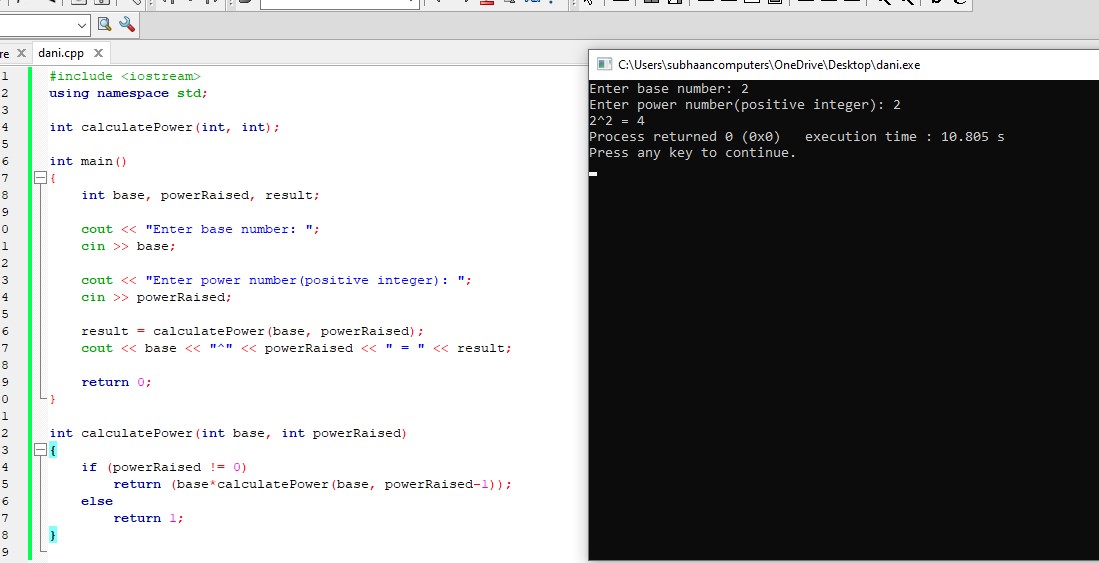
**Task No 05:**

****

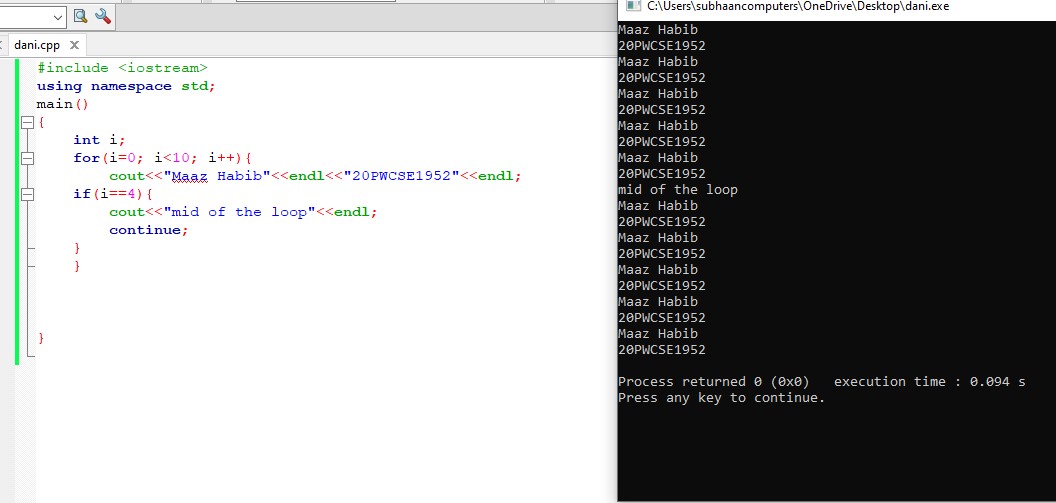
**Task No 06:**

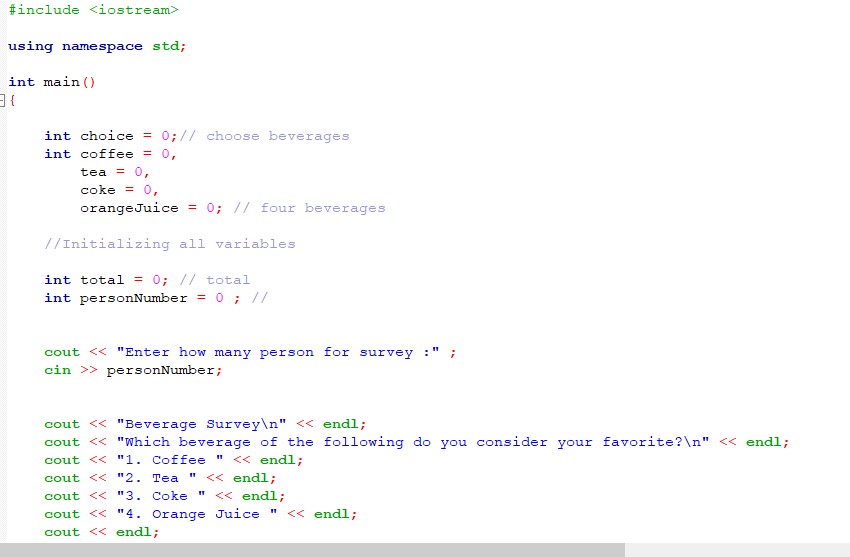
****

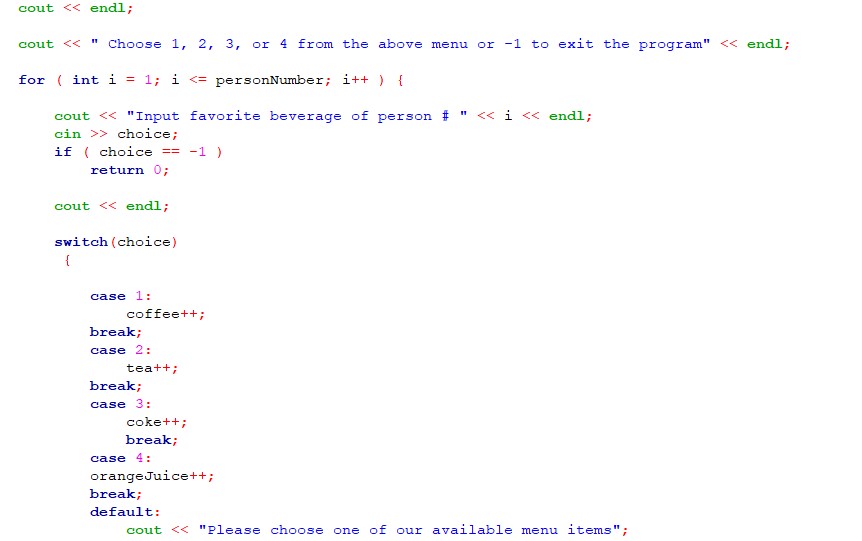
**Task No 07:**

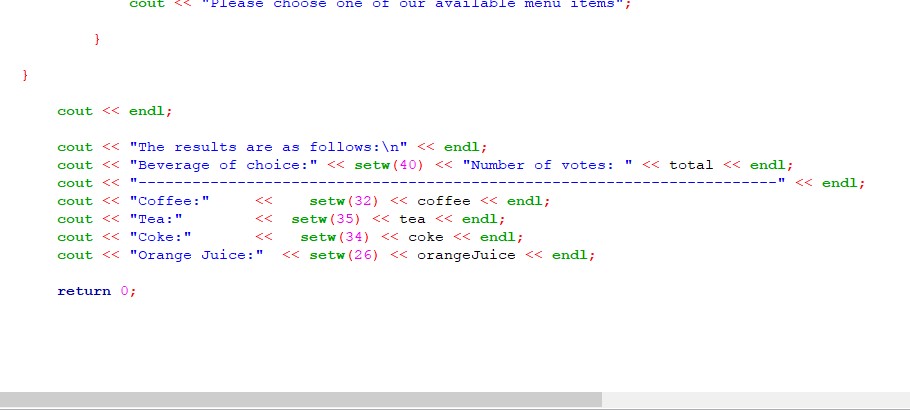
****

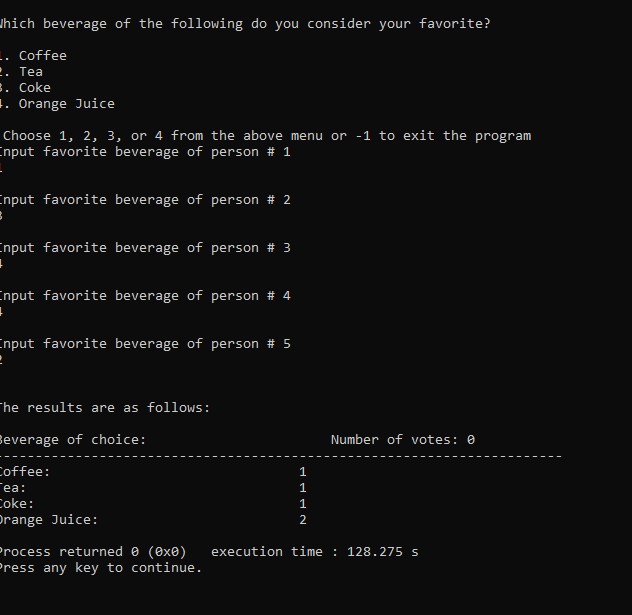
**Task No 08:**

****

**Task No 09** ****

****

****

****