Government College University, Lahore

Department of Computer Science

Programming Fundamentals

Lab – 09

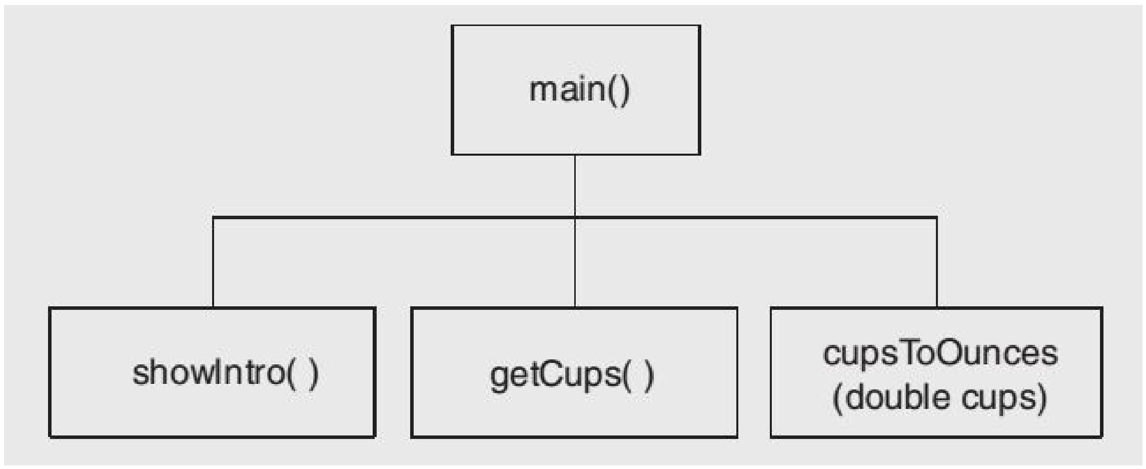
Task-01

Your friend runs a catering company. Some of the ingredients that his recipes required are measured in cups. When he goes to the grocery store to buy those ingredients, however, they are sold only by the fluid ounces. He asked you to write a program that converts cups to fluid ounces.

You design the following algorithm

1. Display an introductory screen that explains what the program does.
2. Get the number of cups
3. Converts the number of cups to fluid ounces and display the result (One cup is equal to 8 fluid ounces)

Find below the hierarchy that how program is broken down into functions



You have to write a complete program based on the designed algorithm.

Task-02 (Monkey Business)

A local zoo wants to keep track of how many pounds of food each of its three monkeys eats each day during a typical week. Write a program that stores this information in a two dimensional 3 X 7 array, where each row represents a different monkey and each column represents a different day of the week. The program should first have the user input the data for each monkey. Then it should create a report that includes the following information:

- Average amount of food eaten per day by the whole family of monkeys.
- Average amount of food eaten per week by each monkey
- The least amount of food eaten during the week by any one monkey.
- The greatest amount of food eaten during the week by any one monkey.

Input Validation: Do not accept negative numbers for pounds of food eaten.

Task-03 (2D array operations)

Write a program that creates a 2D array of integers of size nXn initialized with test data. The program should have the following functions:

- **`getTotal`**. This function should accept a 2D array as an argument and return the total of all the values in the array.
- **`getAverage`**. This function should accept a 2D array as an argument and return the average of all the values in the array.
- **`getRowTotal`**. This function should accept a 2D array as the first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the total of the values in the specified row.
- **`getColumnTotal`**. This function should accept a 2D array as the first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The function should return the total of the values in the specified column.
- **`getHighestInRow`**. This function should accept a 2D array as the first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the highest value in the specified row.

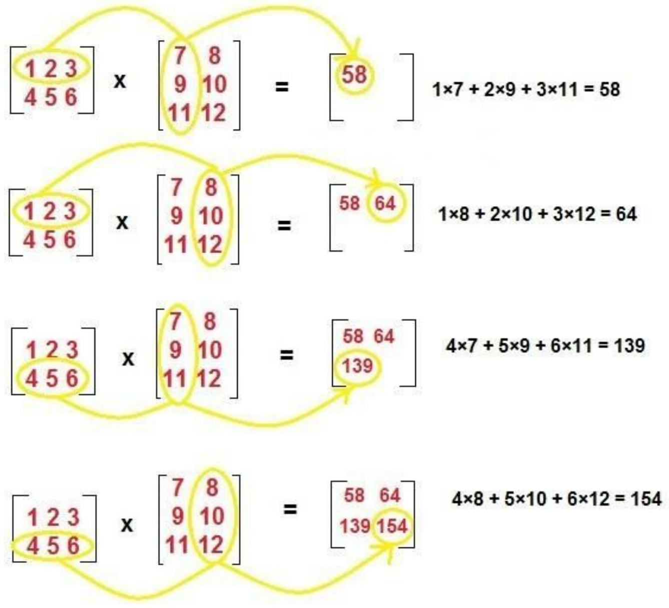
- **`getLowestInRow`**. This function should accept a 2D array as the first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the lowest value in the specified row.

Demonstrate each of the functions in your program using switch structure to design menu for function call.

Task-04 (Matrix Multiplication)

When we do **multiplication**: The number of columns of the 1st **matrix** must equal the number of rows of the 2nd **matrix**. And the result will have the same number of rows as the 1st **matrix**, and the same number of columns as the 2nd **matrix**.

A procedure of matrix multiplication is explained through the below given image.



Write a program that accepts two matrix according the above definition. First ask the user to enter the size of two matrix and validate that the given matrix size is correct for multiplication. Then after multiplication generate a third matrix and print it.

Task-05 (Matrix Addition)

Two **matrices** may be added or subtracted only if they have the same dimension; that is, they must have the same number of rows and columns. **Addition** or subtraction is accomplished by adding or subtracting corresponding elements. For example, consider **matrix A** and **matrix B**.

$$
\begin{aligned}
A + B &= \begin{bmatrix} 2 & 2 & 1 \\ 1 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 5 & 7 & 1 \\ 0 & 3 & 0 \\ 1 & 0 & 8 \end{bmatrix} \\
&= \begin{bmatrix} 2+5 & 2+7 & 1+1 \\ 1+0 & 5+3 & 0+0 \\ 0+1 & 0+0 & 1+8 \end{bmatrix} \\
&= \begin{bmatrix} 7 & 9 & 2 \\ 1 & 8 & 0 \\ 1 & 0 & 9 \end{bmatrix}
\end{aligned}
$$

Write a program that implements the matrix addition.

Task-6 (Transpose of a Matrix)

The **transpose of a matrix** is a new **matrix** whose rows are the columns of the original. ... The superscript "T" means "**transpose**". Another way to look at the **transpose** is that the element at row r column c in the original is placed at row c column r of the **transpose**.

$$
\begin{bmatrix} \boxed{9 \quad 0 \quad 1} \\ 0 \quad 11 \quad 1 \\ 1 \quad 1 \quad 4 \\ 1 \quad 0 \quad 1 \end{bmatrix}
\xrightarrow{\text{Transpose}}
\begin{bmatrix} \boxed{\begin{matrix} 9 \\ 0 \\ 1 \end{matrix}} & \begin{matrix} 0 \\ 11 \\ 1 \end{matrix} & \begin{matrix} 1 \\ 1 \\ 4 \end{matrix} & \begin{matrix} 1 \\ 0 \\ 1 \end{matrix} \end{bmatrix}
$$

Write a program that inputs a matrix and then compute the transpose. The transpose of the matrix must be stored in the new matrix.

Task-07 (Functions calling functions)

Implement the Following Functions

➔ **void get_input(int& input1, int& input2)**

Reads two integers from the keyboard and stores them in input1 and input2 respectively.

➔ **void swap_values(int& variable1, int& variable2)**

swaps the values of variable1 and variable2.

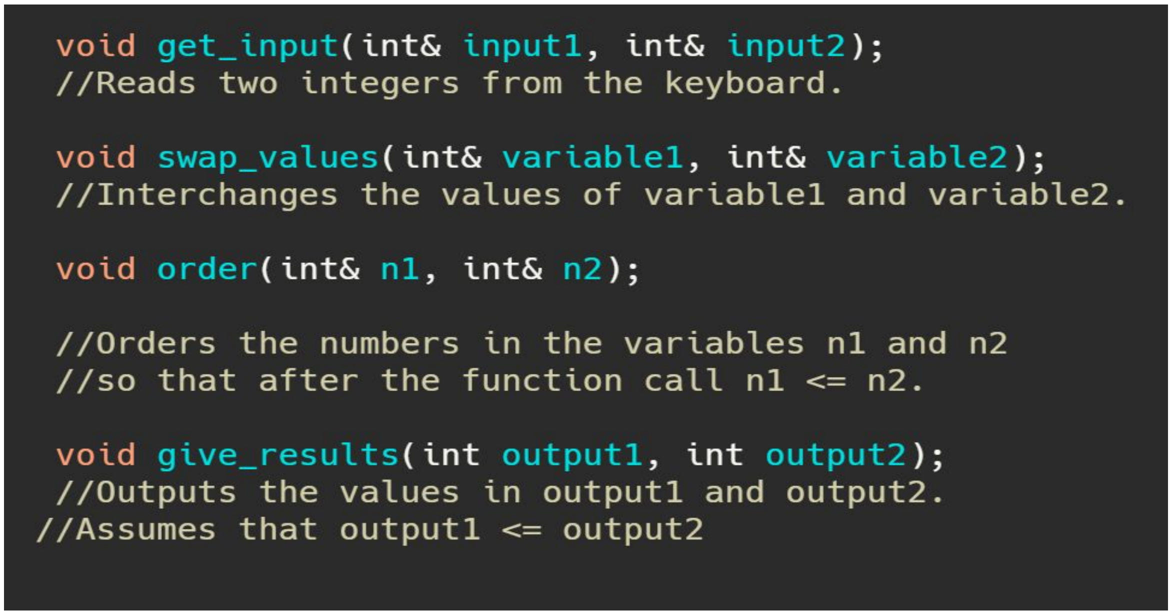
➔ **void order(int& n1, int& n2)**

Orders the numbers in the variables n1 and n2 so that after the function call
n1 <= n2 i.e(swaps the n1 and n2 only if n1<=n2 using **swap_values** function)

➔ **void give_results(int output1, int output2)**

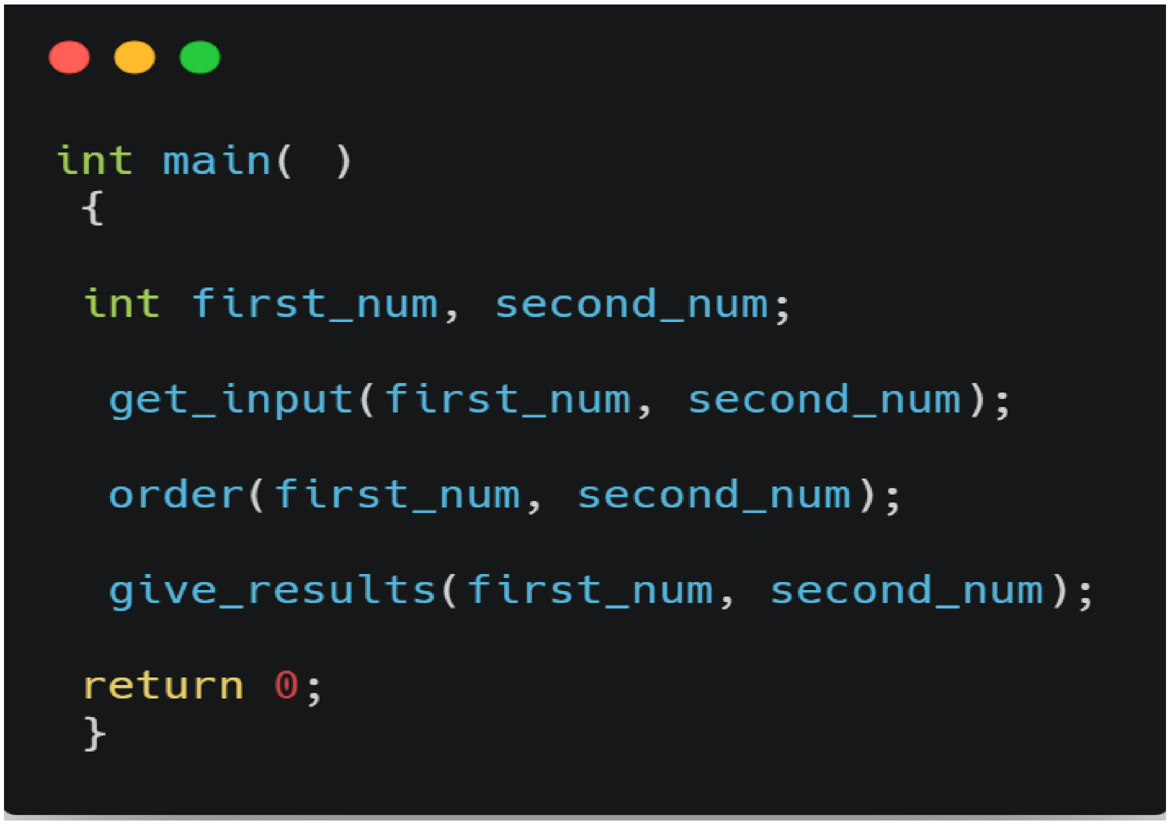
Outputs the values in output1 and output2. Assumes that output1 <= output2

Following are the function declarations:



```
 void get_input(int& input1, int& input2);
 //Reads two integers from the keyboard.

 void swap_values(int& variable1, int& variable2);
 //Interchanges the values of variable1 and variable2.

 void order(int& n1, int& n2);

 //Orders the numbers in the variables n1 and n2
 //so that after the function call n1 <= n2.

 void give_results(int output1, int output2);
 //Outputs the values in output1 and output2.
//Assumes that output1 <= output2
```

after implementing the above functions please test them using following **main function**



```
int main( )
 {

 int first_num, second_num;

  get_input(first_num, second_num);

  order(first_num, second_num);

  give_results(first_num, second_num);

 return 0;
 }
```

Task-08 (Function Overloading)

Create a program named Billing that includes three overloaded computeBill() functions for a store.

•• When computeBill() receives a single parameter, it represents the price of one product ordered. Add 8% tax, and return the total due.

•• When computeBill() receives two parameters, they represent the price of a product and the quantity ordered. Multiply the two values, add 8% tax, and return the total due.

•• When computeBill() receives three parameters, they represent the price of a product , the quantity ordered, and a coupon value. Multiply the quantity and price, reduce the result by the coupon value, and then add 8% tax and return the total due.

Write a main() method that tests all three overloaded methods.

Task-09 (Recursion)

write a function that takes a positive number (do not accept –ve number) as an argument and sum all the numbers from 1 up to the number without using any loop or goto statement.

Task-10 (Factorial)

write a recursive function to find the factorial of a number given as argument.