Develop a calculator program that performs basic arithmetic operations such as addition, subtraction, multiplication, and division. Allow the user to input two numbers and choose an operation to perform.

Algorithm

1) Start
2) Input
3) Operation choice
4) Perform calculations
5) Output results
6) end

```
#include <iostream>
#include <string>
using namespace std;
   int main() {
       double operand1,operand2,result;
        char operation;
        cout<<"enter the first number :";
        cin>>operand1;
        cout<<"enter the second  number :";
        cin>>operand2;
       cout << "Choose operation (+, -, *, /): ";
       cin>>operation;
         switch(operation) {
                case'+':
                result = operand1 + operand2;
                break;
                case'-'
                result = operand1 - operand2;
                break
                case'*':
```

```
            result = operand1 * operand2;
            break;
            case'/':
               if(operand2 != 0){
                  result = operand1/operand2;
               }
               else{
                  cout<<"error:division by zero not allowed"<<endl;
                  return 1;
               }
               break;
            default:
            cout<<"error,invalid operation"<<endl;
            return 1;

      }
            cout << "Result: " << result << endl;
            return 0;
      }
```

Build a simple console-based to-do list manager that allows users to add, view, and delete tasks

Task Input: Allow users to input tasks they want to add to the list.

Add Task: Implement a function to add tasks to the list.

View Tasks: Display the list of tasks with their status (completed or pending).

Mark Task as Completed: Allow users to mark tasks as completed.

Remove Task: Provide an option to remove tasks from the list

```
#include <iostream>
#include <vector>
```

```
struct ToDoTask {
    std::string taskDescription;
    bool isCompleted;

    ToDoTask(const std::string& desc) : taskDescription(desc), isCompleted(false)
{}
};



void addTask(std::vector<ToDoTask>& tasks, const std::string& description) {
    tasks.push_back(ToDoTask(description));
    std::cout << "Task added: " << description << std::endl;
}

void displayTasks(const std::vector<ToDoTask>& tasks) {
    if (tasks.empty()) {
        std::cout << "No tasks available.\n";
    } else {
        std::cout << "Tasks:\n";
        for (size_t i = 0; i < tasks.size(); ++i) {
            std::cout << i + 1 << ". " << tasks[i].taskDescription << " - "
                      << (tasks[i].isCompleted ? "Completed" : "Pending") << "\n";
        }
    }
}
```

```
void completeTask(std::vector<ToDoTask>& tasks, size_t index) {
    if (index >= 1 && index <= tasks.size()) {
        tasks[index - 1].isCompleted = true;
        std::cout << "Task marked as completed: " << tasks[index - 1].taskDescription << std::endl;
    } else {
        std::cout << "Invalid task index.\n";
    }
}

void removeTask(std::vector<ToDoTask>& tasks, size_t index) {
    if (index >= 1 && index <= tasks.size()) {
        std::cout << "Task removed: " << tasks[index - 1].taskDescription << std::endl;
        tasks.erase(tasks.begin() + index - 1);
    } else {
        std::cout << "Invalid task index.\n";
    }
}

int main() {
    std::vector<ToDoTask> tasks;

    while (true) {
        std::cout << "\n===== To-Do List Manager =====\n";
```

```
std::cout << "1. Add Task\n";
std::cout << "2. View Tasks\n";
std::cout << "3. Mark Task as Completed\n";
std::cout << "4. Remove Task\n";
std::cout << "5. Exit\n";
std::cout << "Enter your choice: ";

int choice;
std::cin >> choice;

switch (choice) {
    case 1: {
        std::string taskDescription;
        std::cout << "Enter task description: ";
        std::cin.ignore();  // Clear the input buffer
        std::getline(std::cin, taskDescription);
        addTask(tasks, taskDescription);
        break;
    }
    case 2:
        displayTasks(tasks);
        break;
    case 3: {
        size_t taskIndex;
        std::cout << "Enter the index of the task to mark as completed: ";
        std::cin >> taskIndex;
```

```
            completeTask(tasks, taskIndex);
            break;
        }
        case 4: {
            size_t taskIndex;
            std::cout << "Enter the index of the task to remove: ";
            std::cin >> taskIndex;
            removeTask(tasks, taskIndex);
            break;
        }
        case 5:
            std::cout << "Exiting program.\n";
            return 0;
        default:
            std::cout << "Invalid choice. Please enter a number between 1 and 5.\n";
        }
    }

    return 0;
}
```

Build a simple console-based Tic-Tac-Toe game that allows two players to play against each other TASK 3 TIC-TAC-TOE GAME Game Board:

Create a 3x3 grid as the game board. Players: Assign "X" and "O" to two players.

Display Board: Show the current state of the board. Player Input: Prompt the current player to enter their move.

Update Board: Update the game board with the player ' s move

. Check for Win: Check if the current player has won.

Check for Draw: Determine if the game is a draw. Switch Players: Alternate turns between "X" and "O" players.

Display Result: Show the result of the game (win, draw, or ongoing).

Play Again: Ask if the players want to play another game

```
#include <iostream>
#include <vector>

using namespace std;



void displayBoard(const vector<vector<char>>& board) {
   for (int i = 0; i < 3; ++i) {
      for (int j = 0; j < 3; ++j) {
         cout << board[i][j];
         if (j < 2) cout << " | ";
      }



cout << endl;
      if (i < 2) cout << "---------" << endl;
   }
   cout << endl;
}

bool checkWin(const vector<vector<char>>& board, char player) {
```

```
    for (int i = 0; i < 3; ++i) {
        if (board[i][0] == player && board[i][1] == player && board[i][2] == player)
            return true;
        if (board[0][i] == player && board[1][i] == player && board[2][i] == player)
            return true;
    }

    if (board[0][0] == player && board[1][1] == player && board[2][2] == player)
        return true;
    if (board[0][2] == player && board[1][1] == player && board[2][0] == player)
        return true;

    return false;
}

bool checkDraw(const vector<vector<char>>& board) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (board[i][j] == ' ')
                return false;
        }
    }
    return true;
}
```

```
int main() {
vector<vector<char>> board(3, vector<char>(3, ' '));
char currentPlayer = 'X';

while (true) {
    displayBoard(board);


    int row, col;
    cout << "Player " << currentPlayer << ", enter your move (row and column, both ranging from 0 to 2): ";
    cin >> row >> col;


    if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != ' ') {
        cout << "Invalid move. Try again." << endl;
        continue;
    }


    board[row][col] = currentPlayer;

    if (checkWin(board, currentPlayer)) {
        displayBoard(board);
        cout << "Player " << currentPlayer << " wins!" << endl;
        break;
    }
```

```
    if (checkDraw(board)) {
        displayBoard(board);
        cout << "It's a draw!" << endl;
        break;
    }


    currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
}


char playAgain;
cout << "Do you want to play again? (y/n): ";
cin >> playAgain;

if (playAgain == 'y' || playAgain == 'Y') {

    board = vector<vector<char>>(3, vector<char>(3, ' '));

    currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

} else {
    cout << "Thanks for playing! Goodbye!" << endl;
}
```

```
    return 0;
}
```

Create a program that generates a random number and asks the user to guess it. Provide feedback on whether the guess is too high or too low until the user guesses the correct number

```
#include <iostream>
#include <cstdlib>
#include <ctime>

int main() {

    std::srand(static_cast<unsigned int>(std::time(0)));


    int secretNumber = std::rand() % 100 + 1;

    int userGuess;
    int attempts = 0;

    std::cout << "Welcome to the Number Guessing Game!\n";
    std::cout << "Try to guess the number between 1 and 100.\n";

    do {
        std::cout << "Enter your guess: ";
        std::cin >> userGuess;
```

```
        attempts++;


        if (userGuess == secretNumber) {
            std::cout << "Congratulations! You guessed the correct number in " << attempts << " attempts.\n";
        } else if (userGuess < secretNumber) {
            std::cout << "Too low! Try again.\n";
        } else {
            std::cout << "Too high! Try again.\n";
        }
    } while (userGuess != secretNumber);


    return 0;
}
```