1. Muito simples

    1.1. Primeiro foi acrescentada a biblioteca math para poder fazer as operações necessárias. Dentro da função que calcula a distância foi criada uma variável float, dist, para armazenar o valor e depois foi calculada a distância dos pontos com a fórmula $d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$ e o valor foi salvado na variável dist. Após disso, dist foi retornado.

    1.2. Dentro da função foi criada uma variável float chamada c, o valor da fórmula $\frac{(F-32)\cdot 5}{9} = c$ foi salvado na variável c e foi retornado.

2. Simples

    2.1. Dentro da função foram criadas 5 variáveis int: i, x, y, vetor[10] e result.
    é feito um loop para o usuário inserir os 10 valores do vetor[10], depois o usuário insere os valores para X e Y, é salvado o valor de vetor[x] + vetor[y] em result e por último o result é retornado.

    2.2. É criada uma variável para a soma dos divisores do número, soma_div, e é feito um loop de 1 até o número para pegar todos os divisores do número. Para ver se o número é um divisor ou não, é conferido se o arredondamento para cima do número/i é igual ao arredondamento para baixo.

    É conferido de o número é igual ao soma_div, caso seja igual, 0 é retornado, caso seja diferente, 1 é retornado.

    2.3. É criada uma variável que é a metade do número, caso o número seja 0 ou 1, é retornado 1. É feito um loop que vai de 0 a metade e verifica se tem algum número em que o resto de $\frac{número}{i}$ é 0, é retornado 0, caso não encontre nenhum número com essa condição, é retornado 1;

    2.4. São criadas duas variáveis, uma onde é armazenado o tamanho da string e outra que conta o número de vezes que a letra é lida. um loop percorre a string e toda a vez que a tolower() da letra da string é a mesma que a tolower() da letra escolhida seja igual, é adicionado 1 à variável que conta o número de vezes que a letra foi vista, no final essa variável é retornada.

3.

    3.1. No mesmo código foi colocada a função feita em muito simples 1.
    5 variáveis são criadas, a, b, c, cosseno e ângulo. a é a distância $\overline{AB}$, b é a distância $\overline{BC}$ e c é a distância $\overline{CA}$. A variável cosseno é $\frac{a^2+b^2-c^2}{2\cdot a\cdot b}$ (lei dos cossenos) e o ângulo é o arco cosseno de cosseno. É retornado ângulo.

    3.2. É feita uma função, verifica_num_em_array, que percorre uma array e caso na array tenha o número inserido, ela retorna 1, caso contrário, retorna 0.

São criadas variáveis num_nao_repetidos que tem como valor o tamanho da lista e num_usados. São feitos dois loop dentro do outro de 0 até o tamanho da lista onde, cada vez que lista[i] = lista [j] se j ≠ i, é removido 1 de num_nao_repetidos.

É criada uma array do tamanho de num_nao_repetidos e percorrida a string, para cada elemento da lista é chamada a função verifica_num_em_array que verifica que o elemento da lista está na array, caso não esteja, ele é inserido na array. No final a array é retornada.

4.

4.1. É recebido o input da linha analisada, é verificado se a linha é composta apenas por < > . e a linha é percorrida, cada vez que vem <, é adicionado 1 na variável num_left_side, quando vem >, é adicionado 1 na variável num_right_side e quando vem . é adicionado 1 em sand_grains.

Caso o num_left_side ≥ num_right_side, o num_right_side é salvado em diamante, caso contrário, o num_lefti side é salvado em diamante. É impresso uma frase falando diamantes e sand_grains e é retornado diamantes.

4.2. Eu criei uma função para verificar se o rei está no movimento de cada tipo de peça:

Para o peão, verifica as duas diagonais dependendo da cor da peça;
Para a torre eu verifiquei cada um dos lados até achar uma peça ou acabar o tabuleiro;
Para o bispo eu verifiquei cada diagonal até achar alguma peça ou acabar o tabuleiro;
Para a rainha eu só chamei a função do bispo e da torre, já que o movimento dela é uma junção do movimento dessas duas peças.

Nessas funções, sempre o primeiro passo era identificar qual era o rei que estava sendo procurado, e retornava 1 caso o rei não fosse achado e 0 caso o rei fosse achado, além de imprimir que o rei alvo estava em cheque e qual era a partida que estava sendo analizada.

Também foi feita uma função para verificar se o tabuleiro está vazio, onde ela percorria pelo tabuleiro inteiro e, caso houvesse uma casa que não estivesse vazia, retornava 1, caso percorra o tabuleiro inteiro sem achar peça, retorna 0.

Na função checkmate é onde é verificado se tem um rei em cheque. Primeiramente é criada uma variável para contar quantas partidas já foram feitas e depois é pedido para o usuário inserir um tabuleiro, é verificado se o tabuleiro está vazio, caso esteja vazio o programa encerra, caso contrário o programa continua.

É percorrido o tabuleiro inteiro e, para cada peça detectada, é verificada a peça e é chamada a função da peça encontrada, caso a peça consiga matar o rei, soma um na variável que conta o número de partidas e o programa volta para o início pedindo o tabuleiro para o usuário, caso contrário, o tabuleiro só continua a ser percorrido. Caso nenhuma peça possa matar o rei, é anunciado que nenhum rei está em cheque e o programa volta para o início.

4.3.
4.4.
4.5.