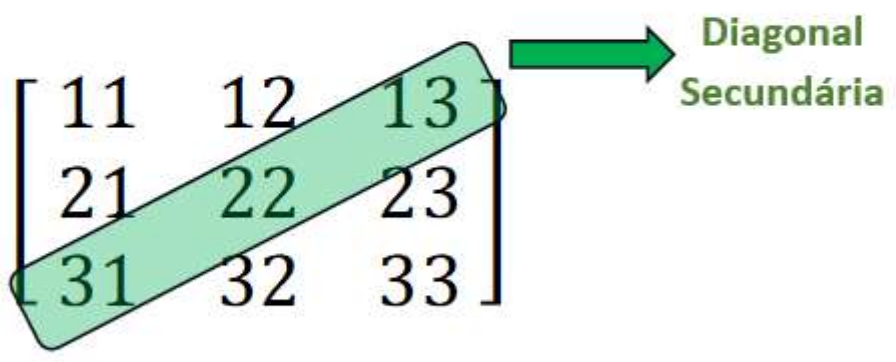
QUESTÃO 01)

Escreva uma função em linguagem C que obtenha os elementos da **Diagonal Secundária** de uma **Matriz Quadrada de Strings** e os armazene em um **Array de Strings**.

Lembre-se de que uma **Matriz Quadrada** é uma matriz em que a quantidade de linhas é igual à quantidade de colunas.

Segue ilustração explicativa de **Diagonal Secundária**:



Neste exemplo a Diagonal Secundária é composta pelos seguintes elementos, nesta ordem:

13
22
31

Segue a assinatura da função que você deve implementar:

```c
typedef char * T_STRING;

//===== ObtemDiagonalSecundaria ================================================//
// Parametros:
//    - T_STRING *in_matriz (input):
//        - Ponteiro para a Matriz Quadrada de Strings
//    - int in_tamanho (input):
//        - Tamanho da Matriz Quadrada. O valor N indica que a Matriz possui N Linhas
//          e N Colunas.
//    - T_STRING *out_diagonal_secundaria (output):
//        - Ponteiro para o Array com os elementos da Diagonal Secundária.
//===============================================================================//
static void ObtemDiagonalSecundaria (T_STRING *in_matriz,
                                     int in_tamanho,
                                     T_STRING *out_diagonal_secundaria) {
    //IMPLEMENTAR ESSA FUNCAO
}
```

A função ***ObtemDiagonalSecundaria*** deve preencher o output ***out_diagonal_secundaria*** com os elementos da Diagonal Secundária da Matriz Quadrada ***in_matriz***.

***** REGRA DE IMPLEMENTAÇÃO *****

Sua solução DEVE utilizar SOMENTE Ponteiros e Aritmética de Ponteiros na manipulação de matriz/array. NÃO DEVEM ser utilizados índices de matriz/array para isso.

Dado o exemplo de código a seguir:

Implemente a função ***ObtemDiagonalSecundaria***.

```c
#include <stdio.h>

typedef char * T_STRING;

static void ImprimeDiagonalSecundaria (T_STRING *in_diagonal_secundaria,
                                       int in_tamanho) {
    // Considere que esta funcao ja esta implementada. Nao precisa implementa-la.
}


//===== ObtemDiagonalSecundaria ================================================//
// Parametros:
//    - T_STRING *in_matriz (input):
//        - Ponteiro para a Matriz Quadrada de Strings
//    - int in_tamanho (input):
//        - Tamanho da Matriz Quadrada. O valor N indica que a Matriz possui N Linhas
//          e N Colunas.
//    - T_STRING *out_diagonal_secundaria (output):
//        - Ponteiro para o Array com os elementos da Diagonal Secundária.
//===============================================================================//
static void ObtemDiagonalSecundaria (T_STRING *in_matriz,
                                     int in_tamanho,
                                     T_STRING *out_diagonal_secundaria) {
    //IMPLEMENTAR ESSA FUNCAO
}

#define TAMANHO 3
int main (void) {
    T_STRING matriz[TAMANHO][TAMANHO] = {
        "Abacaxi", "Banana", "Carambola", "Framboesa", "Goiaba", "Jabuticaba", "Laranja",
        "Manga", "Pitanga"};
    T_STRING diagonal_secundaria[TAMANHO];

    ObtemDiagonalSecundaria((T_STRING *)matriz, TAMANHO, (T_STRING *)diagonal_secundaria);

    ImprimeDiagonalSecundaria((T_STRING *)diagonal_secundaria, TAMANHO);

    return 0;
}
```

QUESTÃO 02)

Desenvolva uma função em C que analise um mapa de bits de 64 posições, numeradas de 1 a 64 e retorne quantos bits estão ligados e suas posições em ordem crescente. Segue a assinatura da função que deve ser implementada:

```
//===== PosicoesBitsLigados ==========================================================//
// Parametros:
//   - unsigned long long int in_mapa_de_bits (input):
//       - Mapa de Bits de 64 posicoes
//   - unsigned int *out_posicoes_bits_ligados (output):
//       - Ponteiro para o Array com as posicoes dos bits ligados
// Retorno:
//   - retorna a quantidade de Bits Ligados
//====================================================================================//
unsigned int PosicoesBitsLigados(unsigned long long int in_mapa_de_bits,
                                 unsigned int *out_posicoes_bits_ligados) {
    //IMPLEMENTAR ESSA FUNCAO
}
```

O mapa de bits a ser analisado é o parâmetro ***in_mapa_de_bits***, que é um inteiro sem sinal de 64 bits. Cada bit de ***in_mapa_de_bits*** representa uma posição do mapa de bits, onde o bit mais significativo de ***in_mapa_de_bits*** representa a posição 1 do mapa de bits e o bit menos significativo de ***in_mapa_de_bits*** representa a posição 64 do mapa de bits.

A função ***PosicoesBitsLigados*** deve analisar o mapa de bits ***in_mapa_de_bits*** e retornar quantos bits estão ligados. Essa função ainda deve preencher o parâmetro de output ***out_posicoes_bits_ligados*** (que é um ponteiro para um array de inteiros sem sinal) com as posições dos bits ligados em ordem crescente.

Exemplo: se o valor de ***in_mapa_de_bits*** for 0x800000000000000C:

8				0				0				0				0				0				0				0				0				0				0				0				0				0				C			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Nesse caso há 3 bits ligados no mapa de bits e suas posições são: 1, 61, 62. Portanto a função ***PosicoesBitsLigados*** deve retornar o valor 3 e o array ***out_posicoes_bits_ligados*** deve ser preenchido com os valores 1, 61, 62 (nessa ordem).

Dado o exemplo de código a seguir:

Implemente a função ***PosicoesBitsLigados***.

```
#include <stdio.h>

//===== PosicoesBitsLigados ==========================================================//
// Parametros:
//   - unsigned long long int in_mapa_de_bits (input):
//       - Mapa de Bits de 64 posicoes
//   - unsigned int *out_posicoes_bits_ligados (output):
//       - Ponteiro para o Array com as posicoes dos bits ligados
// Retorno:
//   - retorna a quantidade de Bits Ligados
//====================================================================================//
unsigned int PosicoesBitsLigados(unsigned long long int in_mapa_de_bits,
                                 unsigned int *out_posicoes_bits_ligados) {
    //IMPLEMENTAR ESSA FUNCAO
}
```

```
int main (void) {
    unsigned long long int mapa_de_bits = 0x800000000000000C;
    unsigned int posicoes_bits_ligados[64];
    unsigned int quantidade_bits_ligados = 0;

    quantidade_bits_ligados = PosicoesBitsLigados(mapa_de_bits, posicoes_bits_ligados);

    printf("Quantidade de bits ligados: %u\n", quantidade_bits_ligados);
    printf("Posicoes dos bits ligados: ");
    for (int i = 0; i < quantidade_bits_ligados; i++) {
        printf("%u", posicoes_bits_ligados[i]);
        printf((i < quantidade_bits_ligados - 1) ? ", " : "");
    }
    printf("\n");

    return 0;
}
```

QUESTÃO 03)

Segue o código fonte de um programa em C, no qual algumas threads utilizam uma função de gravação de LOGs para escrever os LOGs em um arquivo texto. Todas as threads utilizam a mesma função de gravação de LOGs, ou seja, todos os LOGs são gravados num único arquivo.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>
#include <windows.h>
#include <stdlib.h>
#include <conio.h>

#define LOG_FILENAME_BASE "%s_arq_log.txt"

char gStrBufferLog[1000 + 1]; // Buffer para armazenar a string de log
FILE *pLogFile = NULL;
int stopThread = 0;

static char *get_current_time_string_ex (const int format) {
    // A implementacao dessa funcao foi omitida, pois ela
    // eh irrelevante para a analise
}

static char *get_current_time_string (void) {
    // A implementacao dessa funcao foi omitida, pois ela
    // eh irrelevante para a analise
}

int create_logfile (void) {
    char strFilename[100 + 1];
    sprintf(strFilename, LOG_FILENAME_BASE, get_current_time_string_ex(2));
    pLogFile = fopen(strFilename, "w");

    if (pLogFile == NULL) {
        perror("Erro ao criar arquivo de log");
        fclose(pLogFile);
        return 1;
    }

    return 0;
}

void close_logfile (void) {
    if (pLogFile != NULL) {
        fclose(pLogFile);
        pLogFile = NULL;
    }
}

void write_to_logfile (char *szLogText) {
    if (pLogFile == NULL) return;

    // Formata a string de log
    sprintf(gStrBufferLog, "%s:%d|%s", __FILE__, __LINE__, szLogText);

    // Grava a string de log no arquivo de log
    fprintf(pLogFile, "|%s|TID: 0x%08llX|%s|\n", get_current_time_string(), pthread_self(), gStrBufferLog);
    fflush(pLogFile); // Para garantir que a escrita seja imediatamente realizada no arquivo
}

void *thread_function (void *arg) {
    unsigned int counter = 1;
    char strAux[100 + 1];
    pthread_t *pThreadThis = (pthread_t *)arg;
    while (!stopThread) {
        sprintf(strAux, "Linha de LOG gerada pela Thread: 0x%08llX - Chamada %u", *pThreadThis, counter++);
        write_to_logfile(strAux);
        Sleep(50); // Espera de 50 milissegundos
    }
    sprintf(strAux, "Linha de LOG gerada pela Thread: 0x%08llX - Thread encerrada", *pThreadThis);
    write_to_logfile(strAux);
}

#define NUM_THREADS 3
#define CTRL_X_EVENT 24
int main (void) {
    pthread_t threads[NUM_THREADS];

    if (create_logfile() != 0) {
        exit(EXIT_FAILURE);
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        if (pthread_create(&threads[i], NULL, thread_function, (void *)&threads[i]) != 0) {
            perror("Erro ao criar as threads");
            exit(EXIT_FAILURE);
        }
    }

    printf("Pressione Ctrl+X para encerrar.\n");
    // Loop para verificar a entrada de teclado
    while (1) {
        // Verifica se uma tecla foi pressionada
        if (_kbhit()) {
            // Lê a tecla pressionada
            int key = _getch();

            // Verifica se a tecla pressionada foi Ctrl+X
            if (key == CTRL_X_EVENT) {
                printf("Ctrl+X pressionado. Encerrando o programa...\n");
                stopThread = 1;
                break;
            }
        }
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        if (pthread_join(threads[i], NULL) != 0) {
            perror("Erro ao aguardar a thread");
            exit(EXIT_FAILURE);
        }
    }

    close_logfile();

    return 0;
}
```

Executando esse programa notamos um comportamento inesperado no arquivo de LOG gerado, conforme mostrado a seguir:

```
|2024-05-02 07:21:17.068|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 1|
|2024-05-02 07:21:17.068|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000002 - Chamada 1|
|2024-05-02 07:21:17.068|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 1|
|2024-05-02 07:21:17.128|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000002 - Chamada 2|
|2024-05-02 07:21:17.129|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 2|
|2024-05-02 07:21:17.129|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 2|
|2024-05-02 07:21:17.184|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 3|
|2024-05-02 07:21:17.184|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 3|
|2024-05-02 07:21:17.184|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 3|
|2024-05-02 07:21:17.234|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000002 - Chamada 4|
|2024-05-02 07:21:17.234|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 4|
|2024-05-02 07:21:17.234|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 4|
|2024-05-02 07:21:17.289|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 5|
|2024-05-02 07:21:17.289|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 5|
|2024-05-02 07:21:17.289|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 5|
|2024-05-02 07:21:17.344|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000002 - Chamada 6|
|2024-05-02 07:21:17.344|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 6|
|2024-05-02 07:21:17.344|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 6|
|2024-05-02 07:21:17.398|TID: 0x00000001|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000001 - Chamada 7|
|2024-05-02 07:21:17.398|TID: 0x00000002|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000002 - Chamada 7|
|2024-05-02 07:21:17.398|TID: 0x00000003|C:\Inst\main3.c:83|Linha de LOG gerada pela Thread: 0x00000003 - Chamada 7|
```

A linha destacada é apenas uma de muitas inconsistências desse arquivo de LOG. Ela torna evidente que há uma linha de LOG gerada pela thread 0x00000002 que foi gavada pela thread 0x00000001. Isso significa que está ocorrendo um problema de concorrência entre as threads pela função de gravação de LOGs, que está causando inconsistências e perda de LOGs.

A) Explique por que esse problema está acontecendo.
B) Faça o ajuste necessário no código fonte para resolver esse problema.

QUESTÃO 04)

Desenvolva uma função em C para calcular o valor máximo de lucro que poderia ter sido obtido caso você tivesse comprado e vendido ações de uma empresa num determinado período em que os valores das ações são conhecidos.

Segue a assinatura da função que deve ser implementada.

```
//===== MaximoLucro ============================================================//
// Parametros:
//    - unsigned int precos[] (input):
//        - Array com os precos das acoes classificado em ordem cronologica
//    - unsigned int qtdPrecos (input):
//        - Quantidade de elementos no array de precos[]
// Retorno:
//    - retorna o lucro maximo que pode ser obtido comprando e vendendo uma acao
//==============================================================================//
unsigned int MaximoLucro (unsigned int precos[], unsigned int qtdPrecos) {
    //IMPLEMENTAR ESSA FUNCAO
}
```

O array ***precos*** contém os valores de cotação das ações para cada instante monitorado. Considere que os valores contidos no array estão classificados em ordem cronológica. Considere também que você só pode realizar 1 operação de Compra e 1 operação de Venda e, obviamente, a operação de Venda só pode ser realizada num instante posterior à operação de Compra. Última consideração: a Compra e a Vendas das ações só deve ser considerada caso haja lucro. Caso contrário a função ***MaximoLucro*** deve retornar 0, indicando que não foram realizadas as operações de Compra e Venda das ações.

A função ***MaximoLucro*** deve retornar o valor máximo de lucro que poderia ter sido obtido com a Compra e Venda das ações.

Exemplo, se o array ***precos*** tiver os valores {16, 10, 12, 11, 12, 15, 11}:

Valor da Ação	16	10	12	11	12	15	11
Tempo	t0	t1	t2	t3	t4	t5	t6

Onde: t0 < t1 < t2 < ... < t6

O **lucro máximo** que poderia ter sido obtido **seria 5**, **comprando** as ações no **instante t1** pelo **valor 10** e então **vendendo** as ações no **instante t5** pelo **valor 15**.

Dado o exemplo de código a seguir:

Implemente a função ***MaximoLucro***.

```
#include <stdio.h>

//===== MaximoLucro ============================================================//
// Parametros:
//    - unsigned int precos[] (input):
//        - Array com os precos das acoes classificado em ordem cronologica
//    - unsigned int qtdPrecos (input):
//        - Quantidade de elementos no array de precos[]
// Retorno:
//    - retorna o lucro maximo que pode ser obtido comprando e vendendo uma acao
//==============================================================================//
unsigned int MaximoLucro (unsigned int precos[], unsigned int qtdPrecos) {
    //IMPLEMENTAR ESSA FUNCAO
}


int main (void) {
    unsigned int precosAcoes[] = {16, 10, 12, 11, 12, 15, 11};
    unsigned int lucroMax = MaximoLucro(precosAcoes, (sizeof(precosAcoes) / sizeof(precosAcoes[0])));

    printf("Maximo Lucro: %d\n", lucroMax);

    return 0;
}
```

QUESTÃO 05)

O diagrama a seguir mostra a divisão de memória de um programa.

Explique, com suas palavras, cada um dos tipos de memória representados no diagrama, destacando suas principais características. Forneça exemplos de situações em que cada tipo de memória é comumente utilizado. Certifique-se de abordar como cada tipo de memória é alocado, seu propósito e se é gravado em armazenamento permanente.