Name – Utkarsh Yadav

Roll – 23053172

DS Lab Assignment 1

Implement a function that checks if two given strings are anagrams of each other. Ignore spaces and capitalization. For example, "Listen" and "Silent" are anagrams.

```c
//check if two strings are anagram or not

#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <ctype.h>

bool isAnagram(char *w1, char *w2){
  int len1 = strlen(w1);
  int len2 = strlen(w2);

  if(len1 == len2){
    int countW1[26] = {0};
    int countW2[26] = {0};

    for(int i=0; i<len1; i++){
      int lowerW1 = tolower(w1[i]);
      countW1[ lowerW1 - 'a']++;
      int lowerW2 = tolower(w2[i]);
      countW2[ lowerW2 - 'a']++;
    }

    for(int i=0; i<26; i++){
      if(countW1[i] != countW2[i]){
        return false;
      }
    }
     return true;
  } else {
    return false;
  }
}

int main(){

  char w1[50], w2[50];

  printf("Enter string 1: ");
  scanf("%s", w1);

  printf("Enter string 2: ");
  scanf("%s", w2);

  if(isAnagram(w1, w2)){
    printf("%s and %s are anagrams", w1, w2);
  } else{
    printf("Not Anagrams!");
  }
  return 0;
}

```

```
KIIT0001@Utkarsh MINGW64 /d/Learning C/Assig
nments-SD/Assignment 1/anagram
$ gcc anagram.c -o anagram

KIIT0001@Utkarsh MINGW64 /d/Learning C/Assig
nments-SD/Assignment 1/anagram
$ ./anagram
Enter string 1: listen
Enter string 2: silent
listen and silent are anagrams
```

Implement a function to rotate an array of integers by 'k' positions to the right. For example, if the array is [1, 2, 3, 4, 5, 6, 7] and k is 3, the array should be modified to [5, 6, 7, 1, 2, 3, 4].

```c
//Rotate an array by k to the right
#include <stdio.h>

void rotate(int *array, int size, int k){
    for(int i=0; i<k; i++){
      int final = array[size-1];
        for(int j = size-2; j>=0; j--){
          array[j+1] = array[j];
        }
      array[0] = final;
    }
}

int main(){
  int array[] = {1,2,3,4,5};
  int size = sizeof(array)/sizeof(array[0]);
  int k = 2;

  rotate(array, size, k);

  for(int i=0; i<size; i++){
    printf("%d", array[i]);
  }

  return 0;
}
```

```
KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/rotateToRightByK
$ gcc rotate.c -o rotate

KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/rotateToRightByK
$ ./rotate
45123
```

Create a program that intentionally causes a memory leak by allocating memory and not freeing it. Then modify the program to fix the memory leak and explain the changes made.

```c
//Utkarsh Yadav 23053172

#include <stdio.h>
#include <stdlib.h>

void causeMemoryLeak() {
    int* leakyArray = (int*)malloc(10 * sizeof(int));
    if (!leakyArray) {
        fprintf(stderr, "Memory allocation failed\n");
        return;
    }


    for (int i = 0; i < 10; i++) {
        leakyArray[i] = i;
    }

    // if we don't free the allocated space,
    // it'll cause memory leak
    free(leakyArray);
}

int main() {
    for (int i = 0; i < 100; i++) {
        causeMemoryLeak();
    }

    printf("Program finished\n");
    return 0;
}

```

```
KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/memoryleak
$ gcc memoryleak.c -o memoryleak

KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/memoryleak
$ ./memoryleak
Program finished
```

Implement a function to create a dynamic 2D array (matrix) using pointers. Provide functions to fill the matrix with values, print the matrix, and free the allocated memory.

```c
//Utkarsh Yadav 23053172

#include <stdio.h>
#include <stdlib.h>


int** createMatrix(int rows, int cols) {
    int** matrix = (int**)malloc(rows * sizeof(int*));
    if (!matrix) {
        fprintf(stderr, "Memory allocation failed for rows\n");
        return NULL;
    }

    for (int i = 0; i < rows; i++) {
        matrix[i] = (int*)malloc(cols * sizeof(int));
        if (!matrix[i]) {
            fprintf(stderr, "Memory allocation failed for columns\n");
            for (int j = 0; j < i; j++) {
                free(matrix[j]);
            }
            free(matrix);
            return NULL;
        }
    }
    return matrix;
}


void fillMatrix(int** matrix, int rows, int cols) {
    int value = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = value++;
        }
    }
}


void printMatrix(int** matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}


void freeMatrix(int** matrix, int rows) {
    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);
}

int main() {
    int rows;
    int cols;

    printf("Rows: ");
    scanf("%d", &rows);

    printf("Columns: ");
    scanf("%d", &cols);


    int** matrix = createMatrix(rows, cols);
    if (!matrix) {
        return 1;
    }


    fillMatrix(matrix, rows, cols);

    printMatrix(matrix, rows, cols);

    freeMatrix(matrix, rows);

    return 0;
}

```

```
KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/dynamicMatrix
$ gcc dynamicMatrix.c -o dynamic

KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/dynamicMatrix
$ ./dynamic
Rows: 4
Columns: 4
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
```

Write a function that accepts a pointer to an integer array and its size, then uses pointer arithmetic to reverse the array in place.

```c
//Utkarsh Yadav 23053172

#include <stdio.h>


void reverseArray(int *arr, int size) {
    int *start = arr;
    int *end = arr + size - 1;

    while (start < end) {

        int temp = *start;
        *start = *end;
        *end = temp;

        start++;
        end--;
    }
}


void printArray(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int array[] = {1, 2, 3, 4, 5};
    int size = sizeof(array) / sizeof(array[0]);

    printf("Original array:\n");
    printArray(array, size);

    reverseArray(array, size);

    printf("Reversed array:\n");
    printArray(array, size);

    return 0;
}

```

```
KIIT0001@Utkarsh MINGW64 /d/Learning C/Assignments-SD/Assignment 1/reverseArray
$ ./reverse
Original array:
1 2 3 4 5
Reversed array:
5 4 3 2 1
```

I tried to do the rest but couldn’t as I have run out of time and I don’t know how to do them. I’ll make sure to learn and do the rest as soon as possible.