Week 10:Pointers

10 a) Pointers in C Objective: learn to implement the basic functionalities of pointers in C. https://www.hackerrank.com/challenges/pointer-in-c/problem?isFullScreen=true

Source Code:

```
#include <stdio.h>
void update(int *a,int *b) {
    // Complete this function
     int sum,diff;
     sum = *a+*b;
     diff = abs(*a-*b);
     *a = sum;
     *b = diff;
}

int main() {
    int a, b;
    int *pa = &a, *pb = &b;

    scanf("%d %d", &a, &b);
    update(pa, pb);
    printf("%d\n%d", a, b);


    return 0;
}
```

Output:

Input (stdin)

- **4**
- **5**

Your Output (stdout)

- **9**
- **1**

Expected Output

- **9**
- **1**

10 b) Students Marks Sum Objective: Learn using Pointers with Arrays and Functions https://www.hackerrank.com/challenges/students-markssum/problem?isFullScreen=true

Source Code:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

//Complete the following function.

int marks_summation(int* marks, int number_of_students, char gender) {
  //Write your code here.
  int sum = 0;
  for(int i = (gender == 'b' ? 0 : gender == 'g' ? 1 : -
1); i < number_of_students; i+=2) {
    sum += marks[i];
  }
  return sum;
}

int main() {
    int number_of_students;
    char gender;
    int sum;

    scanf("%d", &number_of_students);
    int *marks = (int *) malloc(number_of_students * sizeof (int));

    for (int student = 0; student < number_of_students; student++) {
        scanf("%d", (marks + student));
    }

    scanf(" %c", &gender);
    sum = marks_summation(marks, number_of_students, gender);
    printf("%d", sum);
    free(marks);

    return 0;
}
```

Output:

Input (stdin)

- **3**
- **3**
- **2**
- **5**
- **b**

Your Output (stdout)

- **8**

Expected Output

- **8**

10 c) Sorting Array of Strings Objective: sort a given array of strings into lexicographically increasing order or into an order in which the string with the lowest length appears first. https://www.hackerrank.com/challenges/sorting-array-ofstrings/problem?isFullScreen=true

Source Code:

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

int lexicographic_sort(const char* a, const char* b) {
    return strcmp(a, b);
}

int lexicographic_sort_reverse(const char* a, const char* b) {
    return strcmp(b, a);
}

int sort_by_number_of_distinct_characters(const char* a, const char* b) {
    int count_a = 0, count_b = 0;
    int freq_a[26] = {0}, freq_b[26] = {0};

    for (int i = 0; i < strlen(a); i++) {
        if (freq_a[a[i] - 'a'] == 0) {
            count_a++;
            freq_a[a[i] - 'a'] = 1;
        }
    }

    for (int i = 0; i < strlen(b); i++) {
        if (freq_b[b[i] - 'a'] == 0) {
```

```
            count_b++;
            freq_b[b[i] - 'a'] = 1;
        }
    }

    if (count_a == count_b) {
        return strcmp(a, b);
    } else {
        return (count_a - count_b);
    }
}

int sort_by_length(const char* a, const char* b) {
    int len_a = strlen(a);
    int len_b = strlen(b);

    if (len_a == len_b) {
        return strcmp(a, b);
    } else {
        return (len_a - len_b);
    }
}

void string_sort(char** arr, const int len, int (*cmp_func)(const char* a, const c
har* b)) {
    for (int i = 0; i < len; i++) {
        for (int j = i + 1; j < len; j++) {
            if (cmp_func(arr[i], arr[j]) > 0) {
                char* temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}


int main()
{
    int n;
    scanf("%d", &n);

    char** arr;
```

```
    arr = (char**)malloc(n * sizeof(char*));

    for(int i = 0; i < n; i++){
        *(arr + i) = malloc(1024 * sizeof(char));
        scanf("%s", *(arr + i));
        *(arr + i) = realloc(*(arr + i), strlen(*(arr + i)) + 1);
    }

    string_sort(arr, n, lexicographic_sort);
    for(int i = 0; i < n; i++)
        printf("%s\n", arr[i]);
    printf("\n");

    string_sort(arr, n, lexicographic_sort_reverse);
    for(int i = 0; i < n; i++)
        printf("%s\n", arr[i]);
    printf("\n");

    string_sort(arr, n, sort_by_length);
    for(int i = 0; i < n; i++)
        printf("%s\n", arr[i]);
    printf("\n");

    string_sort(arr, n, sort_by_number_of_distinct_characters);
    for(int i = 0; i < n; i++)
        printf("%s\n", arr[i]);
    printf("\n");
}
```

Output:

Input (stdin)

- **4**
- **wkue**
- **qoi**
- **sbv**
- **fekls**

Your Output (stdout)

- **fekls**
- **qoi**
- **sbv**
- **wkue**

-
  - **wkue**
  - **sbv**
  - **qoi**
  - **fekls**
-
  - **qoi**
  - **sbv**
  - **wkue**
  - **fekls**
-
  - **qoi**
  - **sbv**
  - **wkue**
  - **fekls**
-

Expected Output

- **fekls**
- **qoi**
- **sbv**
- **wkue**
-
  - **wkue**
  - **sbv**
  - **qoi**
  - **fekls**
-
  - **qoi**
  - **sbv**
  - **wkue**
  - **fekls**
-
  - **qoi**
  - **sbv**
  - **wkue**
  - **fekls{-truncated-}**

10.d) Find the sum of a 1D array using malloc()

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

```
int main() {
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    // Dynamically allocate memory for the array
    int *arr = (int *)malloc(n * sizeof(int));

    if (arr == NULL) {
        printf("Memory allocation failed. Exiting...\n");
        return 1; // Exit the program with an error code
    }

    // Input elements into the array
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Calculate the sum of elements
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    printf("Sum of the elements in the array: %d\n", sum);
```

```
    // Free the dynamically allocated memory
    free(arr);

    return 0;
}
```

Output

```
Enter the number of elements in the array: 5
Enter the elements of the array:
1
2
3
4
5
Sum of the elements in the array: 15
```

10.e) Swap two numbers using functions and pointers - call by value and reference.

```
#include <stdio.h>

// Function to swap two numbers using call by value
void swapByValue(int a, int b) {
```

```
    int temp = a;
    a = b;
    b = temp;
}

// Function to swap two numbers using call by reference
void swapByReference(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    // Swap by value
    printf("Before swapping by value: num1 = %d, num2 = %d\n", num1, num2);
    swapByValue(num1, num2);
    printf("After swapping by value: num1 = %d, num2 = %d\n", num1, num2);

    // Swap by reference using pointers
    printf("Before swapping by reference: num1 = %d, num2 = %d\n", num1,
num2);
    swapByReference(&num1, &num2);
```

```
    printf("After swapping by reference: num1 = %d, num2 = %d\n", num1,
num2);


    return 0;
}
```

Output:

Call by value using functions

```
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 10, b = 20
```

Call by Reference using pointers

```
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 20, b = 10
```

10 f) Dynamic Array in C Objective: Handling requests by a Librarian to place the books in the shelves. https://www.hackerrank.com/challenges/dynamic-array-inc/problem?isFullScreen=true

Source Code:

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

/*
 * This stores the total number of books in each shelf.
 */
int* total_number_of_books;

/*
 * This stores the total number of pages in each book of each shelf.
 * The rows represent the shelves and the columns represent the books.
 */
int** total_number_of_pages;
int main()
{
    int total_number_of_shelves;
```

```
    scanf("%d", &total_number_of_shelves);

    int total_number_of_queries;
    scanf("%d", &total_number_of_queries);

    total_number_of_books = (int *)malloc(sizeof(int)*total_number_of_shelves)
;
    total_number_of_pages = (int **)malloc(sizeof(int *)*total_number_of_shelv
es);
    for(int i = 0; i<total_number_of_shelves; i++){
        *(total_number_of_books + i) = 0;
    }
    while(total_number_of_queries--){
        int type_of_query;
        scanf("%d", &type_of_query);
        if(type_of_query == 1){
            int x, y;
            scanf("%d %d", &x, &y);
            int booksInShelf = *(total_number_of_books + x);
            *(total_number_of_pages + x) = (int*)realloc(*(total_number_of_pages
+x),sizeof(int)*(booksInShelf+1));
            *(*(total_number_of_pages+x)+booksInShelf) = y;
            *(total_number_of_books + x) += 1;

        } else if (type_of_query == 2) {
            int x, y;
            scanf("%d %d", &x, &y);
            printf("%d\n", *(*(total_number_of_pages + x) + y));
        } else {
            int x;
            scanf("%d", &x);
            printf("%d\n", *(total_number_of_books + x));
        }
    }

    if (total_number_of_books) {
        free(total_number_of_books);
    }

    for (int i = 0; i < total_number_of_shelves; i++) {
        if (*(total_number_of_pages + i)) {
            free(*(total_number_of_pages + i));
        }
```

```
    }

    if (total_number_of_pages) {
        free(total_number_of_pages);
    }

    return 0;
}
```

Output:

Input (stdin)

- **5**
- **5**
- **1 0 15**
- **1 0 20**
- **1 2 78**
- **2 2 0**
- **3 0**

Your Output (stdout)

- **78**
- **2**

Expected Output

- **78**
- **2**

WEEK-11 : Structure, Union, typedef, bit-fields and enum

11.a) Write a C program to find the total, average of n students using structures

```
#include <stdio.h>

// Define a structure for student information
struct Student {
    char name[50];
    int rollNumber;
    float marks;
```

```
int main() {
    FILE *sourceFile, *destinationFile;
    char sourceFileName[] = "source.txt"; // Change to your source file name
    char destinationFileName[] = "destination.txt"; // Change to your destination file name
    char ch;



    // Open the source file for reading
    sourceFile = fopen(sourceFileName, "r");
    if (sourceFile == NULL) {
        printf("Unable to open the source file.\n");
        return 1;
    }

    // Open the destination file for writing
    destinationFile = fopen(destinationFileName, "w");
    if (destinationFile == NULL) {
        printf("Unable to open the destination file.\n");
        fclose(sourceFile);
        return 1;
    }

    // Copy the contents from source to destination
    while ((ch = fgetc(sourceFile)) != EOF) {
        fputc(ch, destinationFile);
    }
```

```
    printf("File contents copied successfully.\n");

    // Close the files
    fclose(sourceFile);
    fclose(destinationFile);

    return 0;
}
```

Output

File contents copied successfully.

12.d) Merge two files into the third file using command-line arguments.

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

int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("Usage: %s <InputFile1> <InputFile2> <OutputFile>\n", argv[0]);
        return 1;
    }

    FILE *file1, *file2, *outputFile;
    char *inputFileName1 = argv[1];
    char *inputFileName2 = argv[2];
```

```
char *outputFileName = argv[3];
char ch;

// Open the first input file for reading
file1 = fopen(inputFileName1, "r");
if (file1 == NULL) {
    printf("Unable to open the first input file.\n");
    return 1;
}

// Open the second input file for reading
file2 = fopen(inputFileName2, "r");
if (file2 == NULL) {
    printf("Unable to open the second input file.\n");
    fclose(file1);
    return 1;
}

// Open the output file for writing
outputFile = fopen(outputFileName, "w");
if (outputFile == NULL) {
    printf("Unable to open the output file.\n");
    fclose(file1);
    fclose(file2);
    return 1;
}
```

```
    // Copy the contents from the first input file to the output file
    while ((ch = fgetc(file1)) != EOF) {
        fputc(ch, outputFile);
    }

    // Copy the contents from the second input file to the output file
    while ((ch = fgetc(file2)) != EOF) {
        fputc(ch, outputFile);
    }

    printf("Files merged successfully.\n");

    // Close the files
    fclose(file1);
    fclose(file2);
    fclose(outputFile);

    return 0;
}
```

Output

./merge_files inputfile1.txt inputfile2.txt outputfile.txt

12.e) Find no. of lines, words and characters in a file

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

int main() {
    FILE *file;
    char fileName[] = "sample.txt"; // Replace with the file name you want to analyze
    char ch;
    int lines = 0, words = 0, characters = 0;
    int inWord = 0; // Flag to track whether currently in a word

    // Open the file for reading
    file = fopen(fileName, "r");
    if (file == NULL) {
        printf("Unable to open the file for reading.\n");
        return 1;
    }

    while ((ch = fgetc(file)) != EOF) {
        characters++;

        // Count words and lines
        if (ch == '\n') {
            lines++;
        }
```

```
        if (ch == ' ' || ch == '\n' || ch == '\t') {
            inWord = 0;
        } else if (inWord == 0) {
            inWord = 1;
            words++;
        }
    }

    // Close the file
    fclose(file);

    // Display the counts
    printf("Number of lines: %d\n", lines);
    printf("Number of words: %d\n", words);
    printf("Number of characters: %d\n", characters);

    return 0;
}
```

Output:

```
Number of lines: 5
Number of words: 22
Number of characters: 139
```