**CSE 4202-Structured Programming-II Lab. Summer 2022-23**
**Lab Final Examination**
**Date**: April 30, 2024
**Target Group:** 1B
**Topic**: All topics

**Instructions**:

- Task naming format: fullID_T01LF_1B.c
- If you find any issue in problem description/test cases, comment in the google classroom.
- Please comment if you find any tricky test cases that I didn't include and others might forget to handle; please comment! I'll be happy to add.
- I'll mark the modified portions in BLUE color.

**Distribution**:

| | **Lab Task** | **Assignment** |
|---|---|---|
| **1B** | 1, 2, 3, 4, 5 | |

**Task 1:**

One problem with bitwise shifting is that whenever a bit is shifted left or right, the information of that bit is lost and only 0 is used to fill in the new bit. For instance, if you take a 8 bit number like 10000000 and left shift it once, it will become 00000000.

Sometimes, we want to preserve the information in the shifted bit and make the input move like a rotating tape, with cycling. For instance, if you are doing a rotating bit shift and you shift 10000000 2 bits to the left, the new value will be 00000010.

Now, write a program in C that would take a 8 bit number in binary as input alongside a character L or R to depict left and right shift respectively and an integer, the number of shifts required as input and output the shifted 8 bit number with rotation or cycling enabled. Remember, the amount of shift can be more than 8 bits.

| **Sample Input** | **Sample Output** |
|---|---|
| 01000000<br>L 2 | 00000001 |
| 00010000<br>R 7 | 00100000 |
| 00100000<br>R 9 | 00010000 |
| 11110000<br>L 4 | 00001111 |

**Task 2:**

Very often we do not get an exact match of a substring inside another string, but the substring matches to some extent. For instance, if you have a string, (using biological symbols for ease of explanation, it is not accurate) “AAAATCGCATCGA” and you are given a substring string, “CGCAAA” to match, the 2nd substring cannot be directly found under the first string. However, from the 5th index (starting from 0) of the main string, “CGCA” matches from the beginning of the substring. This is the maximum match that is possible, with 4 characters. Other possible matches are just “C” and “CG”.

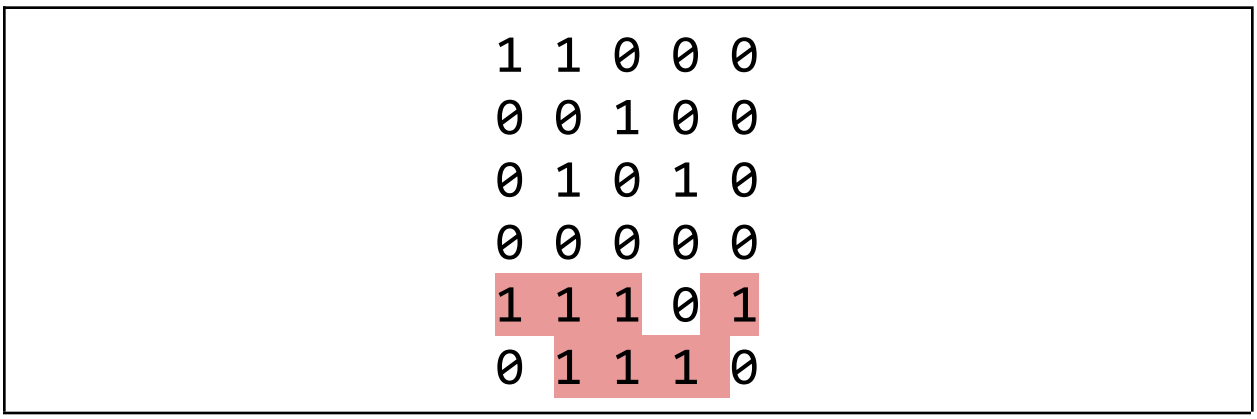
Write a program in C that would take the main string and the substring as input and output 3 things. Maximum matching string, like “CGCA” from the example, number of characters matched, in this case 4 and the index of the main string from where the match started, in this case 5. The strings will not be more than 1024 characters. In case there are multiple maximum matches, output the first one.

| **Sample Input** | **Sample Output** |
|---|---|
| AAAATCGCATCGA<br>CGCAAA | CGCA<br>4 5 |
| AAAATCGCATCGA<br>FX | <br>0 0 |
| AAGGTTCCAATTTTCCCC<br>TCCT | TCC<br>3 5 |
| ABCDEFG<br>ABCDEFG | ABCDEFG<br>7 0 |

**Task 3:**

Mr. Ahmed travels from Abdullahpur to IUT everyday. As everyone knows, the road from Abdullahpur and IUT is full of potholes. To help everyone who travels after, Mr. Ahmed has decided to make a special map of portions of the road from Abdullahpur to IUT.

Mr. Ahmed decided to divide the portion of road into a 2d grid and represent each cell in the grid using 2 values, 0 and 1. 0 means the road is fine there and 1 means there is a pothole. For instance, the following grid represents the condition of a $6 \times 5$ road portion.




Now, you need to write a C program that would take the dimensions of the grid and then the grid itself as input and afterwards, print out the size of the largest pothole in the grid. Two adjacent cells are considered connected if you can travel from one cell to another **vertically, horizontally or diagonally**. The largest pothole in the example grid is highlighted with red and the size of the pothole (cell count) is 7. The road portion will never be larger than $40 \times 40$.

| **Sample Input** | **Sample Output** |
|---|---|
| 2 2<br>0 0<br>0 0 | 0 |
| 6 5<br>1 1 0 0 0<br>0 0 1 0 0<br>0 1 0 1 0<br>0 0 0 0 0 | 7 |

| | |
|---|---|
| 1 1 1 0 1<br>0 1 1 1 0 | |

### Task 4 – Centrist Search

**Problem Statement**

You are given a set of $n$ elements, $a_1$, $a_2$, ..., $a_n$.

Insert all of these $n$ elements in a linked list. Your task is to find the middle element of the linked list. Formally, the middle element is the $(\frac{n}{2} + 1)$th element, where $n$ is the length of the linked list.

**Input**

Each test case will have $n$ positive integers. The input ends when the value 0 is encountered.

**Output**

For each case, print the middle element of the linked list.

**Sample Test Case(s)**

**Input**

```
1 2 3 4 5 6 7 0
```

**Output**

```
Middle Element: 4
```

**Input**

```
1 2 3 4 5 6 7 8 0
```

**Output**

```
Middle Element: 5
```

**Note:** You can traverse the whole linked list **only once**. Don't use $O(n)$ extra memory. Make sure the space complexity is $O(1)$.

# Task 5 — Result Processing System

In Imperial University of Texas (IUT), all courses are of 3.0 credit. In a course, a teacher takes 4 quizzes (out of 15), one mid (out of 75), one final exam (out of 150) in a semester. Among the 4 quizzes, the best 3 are selected. The final mark (100%) is calculated as 10%

converted from overall attendance percentage, 15% from the summation of best 3 quizzes, 25% from mid and 50% from final.

The grading system of IUT is as below.

| Total Marks | Grade |
|---|---|
| 80 to 100 | A |
| 70 to less than 80 | B |
| 60 to less than 70 | C |
| 50 to less than 60 | D |
| Less than 50 | F |

Now write a program in C that would take the data of the students as provided by the teacher as input and print out a sorted list of students, from higher Grade to lower. If the grades of two students are the same, then they will be sorted based on their nickname, lexicographically (A to Z). You can consider that a nickname will not be more than 15 characters and there will be no more than 200 students in the system.

## Input

The rows containing the student details will be provided in the **`marksheet.csv`** file in the following format:

- `<NICKNAME> <ATTENDANCE> <QUIZ1> <QUIZ2> <QUIZ3> <QUIZ4> <MID> <FINAL>`

Note that the **first row** corresponds to the **column headers**. You can download the CSV file using this link: https://drive.google.com/file/d/1GBzn63c3I9HNggbY50rmZPPvazFK9fLy/view?usp=sharing

## Output

Name and letter grade of students printed in sorted order based on the aforementioned rules. Don't forget to include the column headers as well. Name the output file **`gradesheet.csv`**.

| Sample Input (`marksheet.txt`) | Sample Output (`gradesheet.txt`) |
|---|---|

| | |
|---|---|
| NICKNAME,ATTENDANCE,QUIZ1,QUIZ2,QUIZ3,QUIZ4,MID,FINAL<br>Nick,90,15,12,15,10.5,60,145<br>Alphos,70,5,5.5,10,12,40,90<br>Betos,80,6,12,9,13.5,60,100<br>Cron,100,12,12,10.5,13.5,60,120<br>Epsilon,40,5,5,3,7,30,50 | NICKNAME,GRADE<br>Cron,A<br>Nick,A<br>Betos,B<br>Alphos,D<br>Epsilon,F |