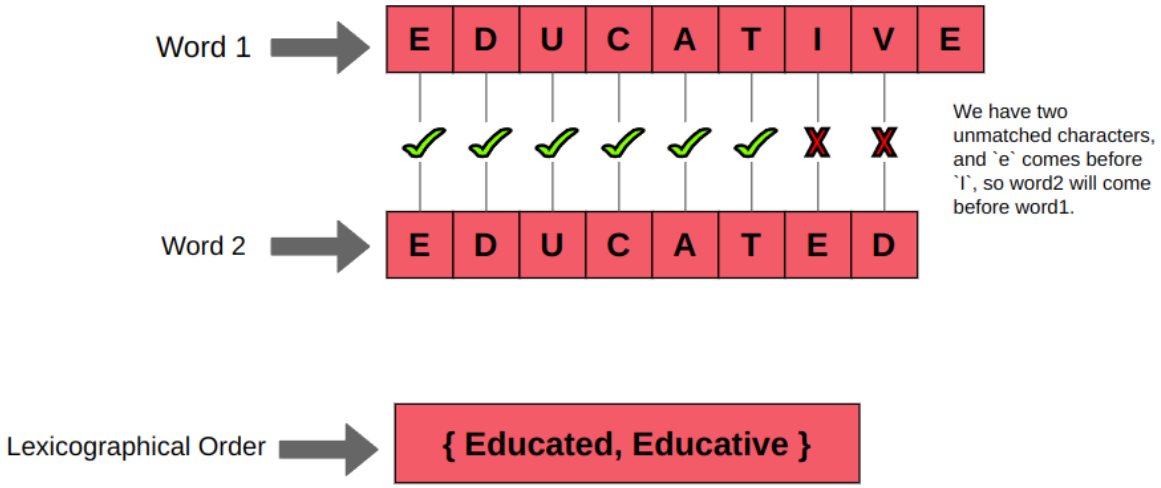
CS101 Lab Quiz 2 - Batch D
19 April 2024 - 20:30 hrs to 23:00 hrs
3 Compulsory Questions - 40 Marks (10, 10, 20)

Instructions:

- Keep your ID card on the table for ready reference. If your ID card isn't with you, you won't be allowed to appear for the quiz/exam.
- Keep your phones, tablets, notes, bags, books, etc., near the instructor's platform.
- Rough sheets will be provided to you.
- **Create a folder on your `Desktop` and name it `submission_YourRollNumber`**
  E.g.: If my roll number is `23k1234` then my folder name is `submission_23k1234`
- **Create all three programs in the newly created folder.**
- Name the program files as mentioned in this pdf only.
- No clarifications will be provided for any question by anyone (TAs/Instructor). When in doubt, make suitable assumptions, state them clearly as comments in your program file itself, and proceed to solve the problem.
- Please note that your answers should NOT include any programming concept that hasn't been covered in the class. You are not allowed to use any advanced concepts of C/C++ like strings, vectors, etc.. Such solutions if found will NOT be graded.
- Marks will be given for each hidden test case that passes.
- At this stage of the course, we expect you to write correct code and fix compilation and logical errors by yourselves. Hence, there will not be any partial marks for any errors. Marks will be given for each hidden test case that passes. Cribs like only one semicolon is missing or instead of <, <= should have been written, etc. will NOT be considered. TAs will not make any changes to your program. It is your responsibility to make it work.
- TAs would be around to help you with respect to the logistics like saving programs, compiling, submitting, etc. They will not help you debug the error or resolve your issues related to syntax/logical errors in your program.
- All the Best!

Filename: LQ02_D_Q1.cpp Sort Words in Dictionary Order (10 Marks)

Write a C++ program to take a positive integer N as input from the user, and then input N words, with each word on a new line. Store the words in a 2D character array. The words will contain only valid lowercase letters i.e. ‘a’ to ‘z’ Sort the words in dictionary/lexicographic order and print them. A **lexicographic order** is an arrangement of characters, words, or numbers in alphabetical order, that is, the letters are sorted from A-Z. Refer to the below example.



Only use commands and concepts covered in class. You are not allowed to use any in-built string functions. If such solutions are found, these will not be considered.

Input Format:
The first line has a positive integer N
The subsequent N lines have N words (one word per line)

Output Format:
Print the sorted words in dictionary order. One word per line

Assume the following:

- The value N entered by the user will be between 1 and 50 (both inclusive)
- The words entered by the user will have valid lowercase letters i.e. ‘a’ to ‘z’.

***Note**:*

- Do not write any C++ statements for printing general messages. For example, the following **should NOT** be present in your program:
    a. **cout << "Enter a number:"**,
    b. **cout << "The computed answer is"**, etc.

  In addition, **do not** print unnecessary spaces unless specified in the program.

Practice Test Cases:

input	output

5
educate
educated
education
educative
educator
	`educate`
	`educated`
	`education`
	`educative`
	`educator`

4 january february march april	april february january march
10 tulip rose iris sunflower orchid daisy peony violet lily daffodil	daffodil daisy iris lily orchid peony rose sunflower tulip violet
8 c python java javascript ruby algol go react	algol c go java javascript python react ruby
8 sparrow jay eagle falcon crow owl	crow eagle falcon hawk jay owl sparrow

swan hawk	swan
15 mountain apple river guitar dream whisper firefly ocean moonlight laughter forest cloud breeze starlight journey	apple breeze cloud dream firefly forest guitar journey laughter moonlight mountain ocean river starlight whisper
10 sunrise whisper river harmony butterfly melody twilight adventure starlight echo	adventure butterfly echo harmony melody river starlight sunrise twilight whisper
7 breeze serenity laughter moonlight blossom tranquil wanderlust	blossom breeze laughter moonlight serenity tranquil wanderlust

Filename: LQ02_D_Q2.cpp Transaction in Bank Account (10 Marks)

This programming assignment has a skeleton code. Do NOT type manually. It is available (just below the test cases in this document, in the Helper code section on Bodhitree and as a .cpp file on the Desktop).

In a bank account, one can perform transactions such as depositing some amount or withdrawing some amount. Based on the type of the transaction, the balance gets updated i.e. gets increased or decreased. If the withdrawal amount is more than the balance, in that case, the withdraw transaction is not allowed.

In this programming assignment, using structures, you need to perform the following transactions.

- Deposit: Deposit some amount
- Withdraw: Withdraw some amount.
    - During withdrawal, if the withdrawal amount is greater than the current balance, then the transaction should not be allowed.
      E.g. Balance: 1100
      Withdraw Rs. 1200.
      This transaction is not allowed, so print “cannot withdraw”

The skeleton code contains:

- The structure and its member variables.
- The main_program that calls some functions.
- Your task is to complete the functions only.
- Do not edit any of the skeleton code that is given to you.

Please go through the comments given in the skeleton code for more details.

Input Format

- The first input is a positive integer N i.e. the number of transactions that will be made
- The second input is a positive integer that denotes the current opening balance
- The next N lines contain (a) a character ‘c’ or ‘d’ denoting credit or debit, followed by a space, followed by (b) the transaction amount.

Constraints:

- Assume that the initial Account Balance will always be greater than or equal to 1000.
- Assume that the value of N i.e. the number of transactions will be between 1 and 10 (both inclusive)
- Assume that the account balance will never exceed 10^9.
- Assume that the value entered by the user for transaction type will be either c or d
- Assume that the value entered for the transaction amount will be between 1 and 10^4 (both inclusive)

Practice Test Cases:

Input	Output
4 1000 d 10000 d 10000 c 10000 d 10001	Initial Balance : 1000 cannot withdraw 1000 cannot withdraw 1000 11000 999 Final Balance : 999
3 1200 c 123 d 124 c 10	Initial Balance : 1200 1323 1199 1209 Final Balance : 1209
5 1200 c 123 d 124 c 10 d 210 c 1	Initial Balance : 1200 1323 1199 1209 999 1000 Final Balance : 1000
2 1000 d 1 c 1	Initial Balance : 1000 999 1000 Final Balance : 1000

2 1000 d 1 d 1	Initial Balance : 1000 999 998 Final Balance : 998
2 1000 c 1 c 1	Initial Balance : 1000 1001 1002 Final Balance : 1002
5 10000 d 1545 c 2123 d 6734 c 4238 c 2999	Initial Balance : 10000 8455 10578 3844 8082 11081 Final Balance : 11081
1 9999 c 1001	Initial Balance : 9999 11000 Final Balance : 11000

Skeleton Code

```
#include<simplecpp>

struct Account {
   int balance;
   int transactions[100];      // the transaction amount
   char transactionType[100];  // c for credit, d for debit
};

int Deposit(Account A,int ith_transaction) {
   // Write the function to
   // credit the ith_transaction made into the account
   // and update the balance
   // return the balance back to the function, performTransactions


}
```

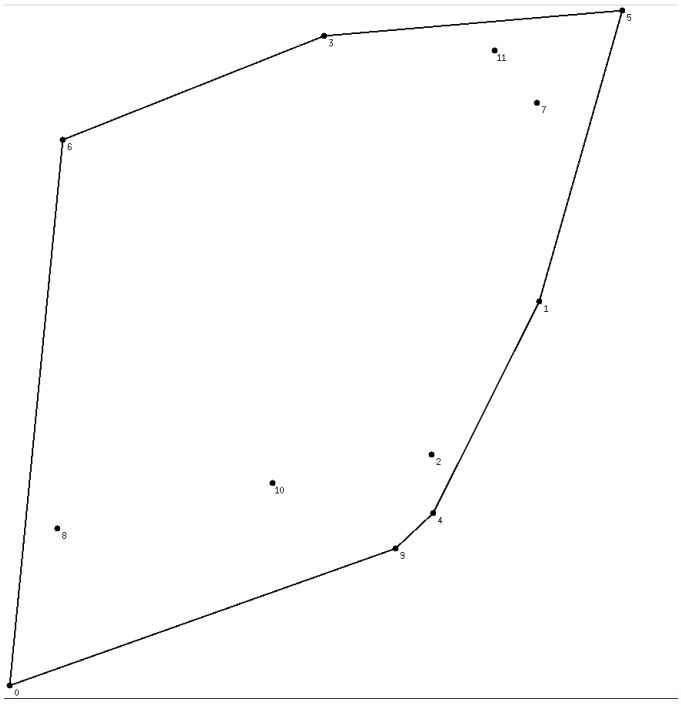
```
int Withdraw(Account A,int ith_transaction){
   // Write the function to
   // debit the ith_transaction made from the account
   // and update the balance
   // return the balance back to the function, performTransactions
   // If the amount being withdrawn is greater than the balance
   // then do not debit, print cannot withdraw,
   // and return the current balance.
   // Basically, that transaction is not done


}

int performTransactions(Account A,int N){
   // For each of the N transactions made, following is done
   // deposit or withdraw accordingly, and update the balance
   // After each credit/debit transaction, print the balance
   // In the end, return the final balance to the main_program
    for(int i=0;i<N;i++){
        if(A.transactionType[i] == 'c'){
            A.balance = Deposit(A,i);
        }
        else{
            A.balance = Withdraw(A,i);
        }
        cout << A.balance << endl;
    }
    return A.balance;
}

main_program
{
   int N; // The number of transactions
   Account A1;
   cin >> N >> A1.balance;

   for(int i = 0; i < N; i++) {
       cin >> A1.transactionType[i];  // c for credit, d for debit
       cin >> A1.transactions[i];   // transaction amount
```

```
    }

    cout<<"Initial Balance : " << A1.balance << endl;

    A1.balance = performTransactions(A1,N);

    cout << "Final Balance : " << A1.balance << endl;
}
```

Filename: LQ02_D_Q3.cpp Convex Hull (20 Marks)

Problem statement

You are given a set of points in a 2D plane. Your task is to implement a function to find the **convex hull** of these points and return the vertices of the convex hull. The convex hull of a set of points is the smallest convex polygon that contains all the points in the set. It's often described as the shape formed by stretching a rubber band around the points. Here is an example with 12 points. The points are shown as dots with numbers (excuse the small font size!) and the line segments constituting the convex hull are also drawn.



Input Format

- The first line of the input contains an integer 'N', the number of points in the set. Assume that 4 <= N <= 30.
- The next N lines of the input contain 2 integers, denoting the x and y coordinates of the points in sequence. The points are indexed 0 through n - 1.

Output Format

Suppose your convex hull has m vertices. Your output should contain the indices of these vertices in sequence, such that they traverse the convex hull in a clockwise fashion. The vertex with the smallest index must be the first in the sequence. It must also be given at the end of the sequence (to draw the m-th line segment back to it). For example, the output for the problem instance in the figure is: **0 6 3 5 1 4 9 0**

To help you with this exercise, we provide two tools.

- First, you are given template code in a file called convex-hull.cpp, which contains a function that uses s++ graphics to plot the points and the convex hull. This function.

  void drawPointsLines(int x[100], int y[100], int N, int seq[100], int m)

  takes the x and y coordinate arrays, the number of points N, the required sequence of points on the convex hull, and the number of points on the convex hull.

- Second, you may use the following idea from coordinate geometry to code up your solution. Consider 2d vectors v1 = (x1, y1) and v2 = (x2, y2). If we have to rotate v1 about the origin until it aligns with v2, do we have to rotate by less than 180 degrees or more? Alternatively, imagine that you have gone on the line from the origin to (x1, y1). Do you now have to turn left or right to go to (x2, y2)? The answer to this question depends on the sign of (x1 * y2 - x2 * y1). In other words, if (x1 * y2 - x2 * y1) is positive, you turn right; if negative, you turn left. Why is the idea relevant? Think of the line segments on the convex hull. What property do they have with respect to all the other points?

Constraints:
4<=N <= 30
The convex hull will not have any collinear points.

SNo	Input	Output
1	4 482 622 245 691 581 522 529 177	0 1 3 2 0
2	8 710 780 8 69 637 857 521 223 160 880 274 866 631 879 708 571	0 6 4 1 3 7 0
3	12 10 883 699 384 559 583 419 39 561 659 807 6 79 174 696 126 72 679 512 705 352 620 641 58	0 6 3 5 1 4 9 0
4	20 664 338 384 153 657 52 297 870 804 81 812 390 574 515 257 282 313 200 842 577 173 235 800 444	2 4 9 12 3 10 18 2

	574 807 695 698 707 134 676 471 472 161 625 230 213 23 200 117	

```
#include <simplecpp>

using std::cin;
using std::cout;

void drawPointsLines(int x[100], int y[100], int N, int seq[100], int
m) {

        initCanvas("Convex hull", 900, 900);

     Text *t1;

        // Plot points
        Circle *c1;
        for(int i=0; i<N; i++) {
            c1 = new Circle(x[i],y[i],4);
            c1->setFill(true);
          t1 = new Text(x[i] + 10, y[i] + 10, i);
        }
        // Draw Lines
        for(int i=1;i<=m;i++){
            Line *l1 = new Line(
x[seq[i-1]],y[seq[i-1]],x[seq[i]],y[seq[i]] );
        }
        wait(20);
}

int main(){

  int n;
  int x[100];
  int y[100];

  cin >> n;
  for(int i = 0; i < n; i++){
```

```
    cin >> x[i];
    cin >> y[i];
  }

  /*
  int seq[];
  int m;
  //Write down code here to set seq[] and m appropriately.
  drawPointsLines(x, y, n, seq, m);
  */

}
```