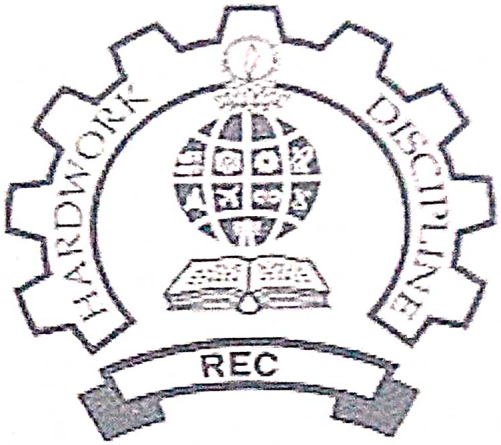
RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution, Affiliated to Anna University
Rajalakshmi Nagar, Thandalam – 602 105



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GE23131 - PROGRAMMING USING C

(Regulation 2023)

LABORATORY MANUAL

Name	: Akash. M
Register No.	: 211624180101
Year / Branch / Section	: I - year / AI & DS / 'A'
Semester	: First
Academic Year	:

Two-Dimensional and Multi-Dimensional Arrays

Ex. No. :

Date :

Add Alternate Elements of 2-Dimensional Array

Problem Statement:

You are given a two-dimensional 3*3 array starting from A [0][0]. You should add the alternate elements of the array and print its sum. It should print two different numbers the first being sum of A 0 0, A 0 2, A 1 1, A 2 0, A 2 2 and A 0 1, A 1 0, A 1 2, A 2 1.

Input Format
First and only line contains the value of array separated by single space.

A00	A01	A02
4	6	9
A10	A11	A12
2	5	8
A20	A21	A22
1	3	7

Output Format
First line should print sum of A 0 0, A 0 2, A 1 1, A 2 0, A 2 2
Second line should print sum of A 0 1, A 1 0, A 1 2, A 2 1

Sample Input
1 2 3 4 5 6 7 8 9

Sample Output
25
20

```
#include<stdio.h>
int main()
{
    int arr[3][3];
    for(int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
        scanf("%d", &arr[i][j]);
    }
    }
    int odd=0, even=0;
    for(int i=0; i<3; i++)
    {
        for (int j=0; j<3; j++)
        {
            if((i+j)%2!=0)
            odd+=arr[i][j];
            else
            even+=arr[i][j];
        }
    }
    printf("%d\n%d", even, odd);
}
```

Ex. No. : Date :

The Wealthy Landlord

Problem Statement:

Shyam Lal, a wealthy landlord from the state of Rajasthan, being an old fellow and tired of doing hard work, decided to sell all the farmland and to live rest of the life with that money. No other farmer is rich enough to buy all the land so he decided to partition the land into rectangular plots of different sizes with different cost per unit area. So he sold these plots to the farmers but made a mistake, being illiterate, he made partitions that could be overlapping. When the farmers came to know about it, they ran to him for compensation of extra money they paid to him. So he decided to return all the money to the farmers of that land which was overlapping with other farmer's land to settle down the conflict. All the portion of conflicted land will be taken back by the landlord.

To decide the total compensation, he has to calculate the total amount of money to return back to farmers with the same cost they had purchased from him. Suppose Shyam Lal has a total land area of 1000 x 1000 equal square blocks where each block is equivalent to a unit square area which can be represented on the co-ordinate axis. Now find the total amount of money, he has to return to the farmers. Help Shyam Lal to accomplish this task.

Input Format:The first line of the input contains an integer N, denoting the total land pieces he had distributed. Next N line contains the 5 space separated integers (X1, Y1), (X2, Y2) to represent a rectangular piece of land, and cost per unit area C.

(X1, Y1) and (X2, Y2) are the locations of bottom and top square block on the diagonal of the rectangular region.

Output Format:

Print the total amount he has to return to farmers to solve the conflict.

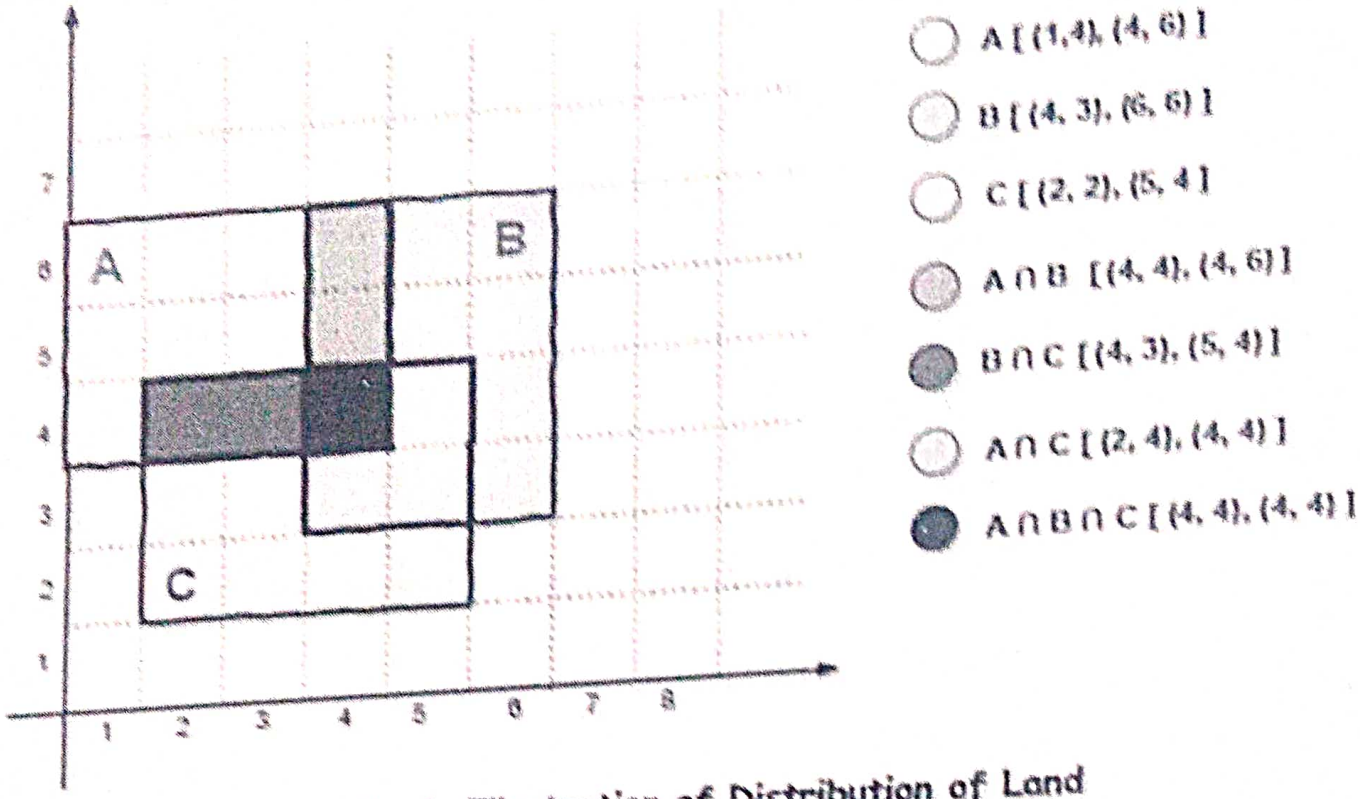
Constraints:

$1 \leq N \leq 100$
$1 \leq X1 \leq X2 \leq 1000$
$1 \leq Y1 \leq Y2 \leq 1000$
$1 \leq C \leq 1000$

Sample Input
3
1 4 4 5 1
4 3 6 5 2
2 2 5 4 3

Sample Output
35

Explanation



Simple Illustration of Distribution of Land

For given sample input (see given graph for reference), compensation money for different
farmers is as follows:

Farmer with land area A: C1 = 5 * 1 = 5
Farmer with land area B: C2 = 6 * 2 = 12
Farmer with land area C: C3 = 6 * 3 = 18

Total Compensation Money = C1 + C2 + C3 = 5 + 12 + 18 = 35

Program:

```
#include<stdio.h>
int main()
{
    int i,j,n,x1,x2,y1,y2,t=0;
    long long int arr[1001][1001]={0};
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d %d %d %d",&x1,&y1,&x2,&y2);
        for(i=x1;i<=x2;i++)
        {
            for(j=y1;j<=y2;j++)
            {
                if(arr[i][j]==0)
                arr[i][j]+=t;
                else if(arr[i][j]>0)
                arr[i][j]=(-1)*(arr[i][j]+t);
                else if(arr[i][j]<0)
                arr[i][j]-=t
            }
        }
    }
    for(i=1;i<1001;i++)
    {
        for(j=1;j<1001;j++)
        {
            if(arr[i][j]<0)
```

```
            total += coin[J[i]];
        }
    }
    printf("%d\n", (-1) * total);
    return 0;
}
```

Ex. No. : Date :

Priority Interview

Problem Statement:

Microsoft has come to hire interns from your college. N students got shortlisted out of which few were males and a few females. All the students have been assigned talent levels. Smaller the talent level, lesser is your chance to be selected. Microsoft wants to create the result list where it wants the candidates sorted according to their talent levels, but there is a catch. This time Microsoft wants to hire female candidates first and then male candidates. The task is to create a list where first all-female candidates are sorted in a descending order and then male candidates are sorted in a descending order.

Input Format
The first line contains an integer N denoting the number of students. Next, N lines contain two space-separated integers, ai and bi. The first integer, ai will be either 1(for a male candidate) or 0(for female candidate). The second integer, bi will be the candidate's talent level.
Constraints: 1 <= N <= 105, 0 <= ai <= 1, 1 <= bi <= 109

Output Format
Output space-separated integers, which first contains the talent levels of all female candidates sorted in descending order and then the talent levels of male candidates in descending order.

Sample Input
5
0 3
1 6
0 2
0 7
1 15
Sample Output
7 3 2 15 6

Program:

```
#include <stdio.h>
struct data
{
   int gen ; int tal ;
};
int main()
{
    int n;
    scanf("%d", &n);
    struct data a[n];
    for (int i=0;i<n; i++)
     scanf ("%d%d",&a[i].gen , &a[i].tal);
    for (int i=0 ;i<n-1 ; i++)
    {
       for (int j=0, j <n-i-1 ; ++ j)
       {
          if (a[j].tal <a [j+1]. tal)
          {
                struct data temp=a[j];
                a[j] = a[j+1] ;
                a[j+1]= temp ;
          }
       }
    }
    for (int i= 0; i<n ;i++)
    {
        if (a[i]. gen ==0)
```

```
        printf("%d", a[i].data);
    for(int i=0; i<n; ++i)
    {
        if(a[i].gen == 1)
        printf("%d", a[i].data);
    }
}
```

Result:

Hence the C program is implemented and executed successfully.

Character Arrays

Ex. No. : Date :

Strings

Problem Statement:

Input Format
You are given two strings, a and b, separated by a new line. Each string will consist of lower-case Latin characters ('a'-'z').

Output Format
In the first line print two space-separated integers, representing the length of a and b respectively.
In the second line print the string produced by concatenating a and b (a + b).
In the third line print two strings separated by a space, a' and b'. a' and b' are the same as a and b, respectively, except that their first characters are swapped.

Sample Input
abcd
ef

Sample Output
4 2
abcdef
ebcd af

Explanation
a = "abcd"
b = "ef"
|a| = 4
|b| = 2
a + b = "abcdef"
a' = "ebcd"
b' = "af"

Program:

```
#include <stdio.h>
int main()
{
    char str1[10], str2[10], t;
    int i=0, j=0;
    int count1=0, count2=0;
    scanf("%s", str1);
    scanf("%s", str2);
    while(str1[i]!='\0')
    {
        count1++;
        i++;
    }
    while(str2[j]!='\0')
    {
        count2++;
        j++;
    }
    printf("%d %d\n", count1, count2);
    printf("%s %s\n", str1, str2);
    t = str1[0];
    str1[0] = str2[0];
    str2[0] = t;
    printf("%s %s", str1, str2);
    return 0;
}
```

Ex. No. :

Date :

Printing Tokens

Problem Statement:

Given a sentence, s, print each word of the sentence in a new line.

Input Format
The first and only line contains a sentence, s.

Constraints
$1 \leq len(s) \leq 1000$

Output Format
Print each word of the sentence in a new line.

Sample Input
This is C

Sample Output
This
is
C

Explanation
In the given string, there are three words ["This", "is", "C"]. We have to print each of these words in a new line.

Hint
Here, once you have taken the sentence as input, we need to iterate through the input, and keep printing each character one after the other unless you encounter a space. When a space is encountered, you know that a token is complete and space indicates the start of the next token after this. So, whenever there is a space, you need to move to a new line, so that you can start printing the next token.

Program:

```
#include<stdio.h>
int main(){
    char s[100];
    scanf("%[^\n]s", s);
    for(int i=0; s[i]!='\0'; i++){
        if(s[i]!=' ')
        printf("%c",s[i]);
        else
        printf("\n");
    }
    return 0;
}
```

Ex. No. :

Date :

Digit Frequency

Problem Statement:

Given a string, s, consisting of alphabets and digits, find the frequency of each digit in the given string.

Input Format
The first line contains a string, num which is the given number.

Constraints
$1 \leq len(num) \leq 1000$
All the elements of num are made of English alphabets and digits.

Output Format
Print ten space-separated integers in a single line denoting the frequency of each digit from 0 to 9.

Sample Input 0
a11472o5t6

Sample Output 0
0 2 1 0 1 1 1 1 0 0

Explanation 0
In the given string:

- 1 occurs two times.
- 2, 4, 5, 6 and 7 occur one time each.
- The remaining digits 0, 3, 8 and 9 don't occur at all.

Hint:

- Declare an array, freq of size 10 and initialize it with zeros, which will be used to count the frequencies of each of the digit occurring.
- Given a string, s, iterate through each of the character in the string. Check if the current character is a number or not.
- If the current character is a number, increase the frequency of that position in the freq array by 1.
- Once done with the iteration over the string, s, in a new line print all the 10 frequencies starting from 0 to 9, separated by spaces.

Program:

```
#include<stdio.h>
int main()
{
    char str[1000];
    scanf("%s", str);
    int hash[10]={0,0,0,0,0,0, 0,0,0,0};
    int temp;
    for(int i=0; str[i] != '\0'; i++)
    {
        temp = str[i] - '0';
        if(temp<=9 && temp>=0)
        {
            hash[temp]++;
        }
    }
    for (int i=0, i<=9; i++)
    {
        printf("%d", hash[i]);
    }
    return 0;
}
```

Ex. No. :

Date :

Monk Takes a Walk

Problem Statement:

Today, Monk went for a walk in a garden. There are many trees in the garden and each tree has an English alphabet on it. While Monk was walking, he noticed that all trees with vowels on it are not in good state. He decided to take care of them. So, he asked you to tell him the count of such trees in the garden.

Note: The following letters are vowels: 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o' and 'u'.

Input Format:
The first line consists of an integer T denoting the number of test cases.
Each test case consists of only one string, each character of string denoting the alphabet (may be lowercase or uppercase) on a tree in the garden.

Output Format:
For each test case, print the count in a new line.

Constraints:
$1 \leq T \leq 10$
$1 \leq$ length of string ≤ 105

Sample Input
2
nBBZLaosnm
JHkIsnZtTL

Sample Output
2
1

Explanation
In test case 1, a and o are the only vowels. So, count=2
Brief Description: Given a string S you have to count number of vowels in the string.

Solution 1:
For each vowel, count how many times it is appearing in the string S. Final answer will the sum of frequencies of all the vowels.

Solution 2:
Iterate overall all the characters in the string S and use a counter (variable) to keep track of number of vowels in the string S. While iterating over the characters, if we encounter a vowel, we will increase the counter by 1.

Time Complexity: O(N) where N is the length of the string S. Space Complexity: O(N)

Program:

```
#include<stdio.h>
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        char str[100000];
        int count=0;
        scanf("%s",str);
        for(int i=0;str[i]!='\0';i++)
        {
            char c = str[i];
            if((c=='a')||(c=='e')||(c=='i')||(c=='o')||(c=='u')
              ||(c=='A')||(c=='E')||(c=='I')||(c=='O')||(c=='U'))
            count++;
        }
    }
    return 0;
}
```

Result:

Thus, the C programs are implemented and executed successfully.

String Handling Function

Ex. No. :

Date :

What is your mobile number?

Problem Statement:

These days Bechan Chacha is depressed because his crush gave him list of mobile number some of them are valid and some of them are invalid. Bechan Chacha has special power that he can pick his crush number only if he has valid set of mobile numbers. Help him to determine the valid numbers.

You are given a string "S" and you have to determine whether it is Valid mobile number or not. Mobile number is valid only if it is of length 10 , consists of numeric values and it shouldn't have prefix zeroes.

Input Format:
First line of input is T representing total number of test cases.
Next T line each representing "S" as described in in problem statement.

Output Format:
Print "YES" if it is valid mobile number else print "NO".
Note: Quotes are for clarity.

Constraints:
1<= T <= 103
sum of string length <= 105

Sample Input
3
1234567890
0123456789
0123456.87

Sample Output
YES
NO
NO

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int flag=1;
        char s[100000];
        scanf("%s",s);
        int k= strlen(s);
        if (k==10)
        {
            for(int i=0;i<10;i++)
            {
                if(s[0]=='0')
                {
                    flag=0;
                    break;
                }
                if (s[i]<'0'||s[i]>'9')
                {
                    flag=0;
                    break;
                }
            }
        }
```

```
        else
        flag = 0;
        if (flag == 1)
        printf("YES\n");
        else
        printf("NO\n");
    }
    return 0;
}
```

Ex. No. :

Date :

Alice and Strings

Problem Statement:

Two strings A and B comprising of lower-case English letters are compatible if they are equal or can be made equal by following this step any number of times:

- Select a prefix from the string A (possibly empty), and increase the alphabetical value of all the characters in the prefix by the same valid amount. For example, if the string is xyz and we select the prefix xy then we can convert it to yx by increasing the alphabetical value by 1. But if we select the prefix xyz then we cannot increase the alphabetical value.

Your task is to determine if given strings A and B are compatible.

Input format
First line: String A
Next line: String B

Output format
For each test case, print YES if string A can be converted to string B, otherwise print NO.

Constraints
$1 \leq \text{len}(A) \leq 1000000$
$1 \leq \text{len}(B) \leq 1000000$

Sample Input
abaca
cdbda

Sample Output
YES

Explanation
The string abaca can be converted to bcbda in one move and to cdbda in the next move.

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[1000000], str2[1000000];
    int flag=1;
    scanf("%s", str1);
    scanf("%s", str2);
    int a=strlen(str1);
    int b=strlen(str2);
    if(a==b)
    {
        for(int i=a-1;i>=0;i--)
        {
            while(str1[i]!=str2[i])
            {
                for(int j=0;j<=1;j++)
                {
                    if(str1[j]<'z')
                    str1[j]++;
                    else
                    {
                        flag=0;
                        break;
                    }if(flag==0)
                    break;
```

```
        }
    }
}
else
flag = 0;
if (flag == 0)
printf ("NO");
else
printf ("YES");
return 0;
}
```

Ex. No. : Date :

Pizza Confusion

Problem Statement:

Joey loves to eat Pizza. But he is worried as the quality of pizza made by most of the restaurants is deteriorating. The last few pizzas ordered by him did not taste good :(. Joey is feeling extremely hungry and wants to eat pizza. But he is confused about the restaurant from wherehe should order. As always he asks Chandler for help.

Chandler suggests that Joey should give each restaurant some points, and then choose the restaurant having maximum points. If more than one restaurant has same points, Joey can choose the one with lexicographically smallest name.

Joey has assigned points to all the restaurants, but can't figure out which restaurant satisfies Chandler's criteria. Can you help him out?

Input Format:
First line has N, the total number of restaurants.
Next N lines contain Name of Restaurant and Points awarded by Joey, separated by a space.
Restaurant name has no spaces, all lowercase letters and will not be more than 20 characters.

Output Format:
Print the name of the restaurant that Joey should choose.

Constraints:
1 <= N <= 105
1 <= Points <= 106

Sample Input
3
Pizzeria 108
Dominos 145
Pizzapizza 49

Sample Output
Dominos

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
    int n;
    scanf("%d",&n);
    char res[n][21];
    int rate[n];
    {
        scanf("%s",res[i]);
        scanf("%d",&rate[i]);
    }
    int max=rate[0];
    char ans[20];
    strcpy(ans,res[0]);
    for(int i=1;i<n;i++)
    {
        if(rate[i]>max)
        {
            max=rate[i];
            strcpy(ans,res[i]);
        }
        else if(rate[i]==max)
        {
            if(strcmp(res[i],ans)<0)
        }   strcpy(ans,res[i]);
```

```
    }
    printf("%s", ans);
    return 0;
}
```

sult:

Thus, the program is successfully implement
d executed successfully

Ex. No. : Date :

Password

Problem Statement:

Danny has a possible list of passwords of Manny's facebook account. All passwords length is odd. But Danny knows that Manny is a big fan of palindromes. So, his password and reverse of his password both should be in the list.

You have to print the length of Manny's password and it's middle character.

Note: The solution will be unique.

Input Format
The first line of input contains the integer N, the number of possible passwords.
Each of the following N lines contains a single word, its length being an odd number greater than 2 and lesser than 14. All characters are lowercase letters of the English alphabet.

Output Format
The first and only line of output must contain the length of the correct password and its central letter.

Constraints
$1 \leq N \leq 100$

Sample Input
4
abc
def
feg
cba

Sample Output
3 b

Program:

```
#include<stdio.h>
#include<string.h>
int main(){
    int n, flag=0;
    char temp;
    scanf("%d", &n);
    char words[n][14];
    for(int i=0;i<n;i++)
    scanf("%s", words[i]);
    char reverse[14];
    for(int i=0;i<n-1;i++)
    {
        strcpy(reverse, words[i]);
    int size = strlen(reverse);
    for(int k=0;k<size/2;k++)
    {
        temp = reverse[k];
        reverse[k] = reverse[size-k-1];
        reverse[size-k-1] = temp;
    }
    for(int j=i+1;j<n;j++)
    {
        if(strcmp(reverse, words[j])==0)
        {
            flag=1;
            break;
        }
    }
```

```
        if (flag == 1)
        break;
    }
    int len = strlen(reverse);
    printf("%d %c", len, reverse[len-2]);
    return 0;
}
```

User-defined Functions & Recursive Functions

Ex. No. : Date :

Find the Factor

Problem Statement:

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the pth element of the list, sorted ascending. If there is no pth element, return 0.

Example
n = 20
p = 3
The factors of 20 in ascending order are {1, 2, 4, 5, 10, 20}. Using 1-based indexing, if p = 3, then 4 is returned. If p > 6, 0 would be returned.

Function Description
Complete the function pthFactor in the editor below.

pthFactor has the following parameter(s):
int n: the integer whose factors are to be found
int p: the index of the factor to be returned

Returns:
int: the long integer value of the pth integer factor of n, or if there is no such factor at that index, then 0 is returned

Constraints
$1 \leq n \leq 1015$
$1 \leq p \leq 109$

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.
The first line contains an integer n, the number to factor.
The second line contains an integer p, the 1-based index of the factor to return.

Sample Input

STDIN		Function
10	→	n = 10
3	→	p = 3

Sample Output
5

Explanation
Factoring n = 10 results in {1, 2, 5, 10}. Return the p = 3rd factor, 5, as the answer.

Program:

```
long pthfactor(long n, long p)
{
    int count = 0;
    for (long i=1 ; i<=n; ++i)
    {
        if (n % i == 0)
        {
            count ++ ;
            if(count == p)
            {
                return i;
            }
        }
    }
    return 0;
}
```

Ex. No. : **Date :**

Prime or Not?

Problem Statement:

Given an integer, if the number is prime, return 1. Otherwise return its smallest divisor greater than 1.

Example
n = 24
The number 24 is not prime: its divisors are [1, 2, 3, 4, 6, 8, 12, 24]. The smallest divisor greater than 1 is 2.

Function Description
Complete the function isPrime in the editor below.

isPrime has the following parameter(s):
long n: a long integer to test

Returns
int: if the number is prime, return 1; otherwise returns the smallest divisor greater than 1

Constraints
$2 \leq n \leq 1012$

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.
The only line of input contains the long integer to analyze, n.

Sample Input

STDIN		Function
2	→	n = 2

Sample Output
1

Explanation
As 2 is a prime number, the function returns 1.

Program:

```
int fourthBit(int number)
{
    int binary[32];
    int i=0;
    while(number>0)
    {
        binary[i]=number%2;
        number/=2;
        i++;
    }
    if(i>=4)
    {
        return binary[3];
    }
    else
    return 0;
}
```

Passing Arrays and Strings to Functions

Ex. No. : Date :

Balanced Array

Problem Statement:

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example: arr=[1,2,3,4,6]

- the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Function Description: Complete the function balancedSum in the editor below.

balancedSum has the following parameter(s): int arr[n]: an array of integers

Returns: int: an integer representing the index of the pivot

Constraints

- $3 \leq n \leq 105$
- $1 \leq arr[i] \leq 2 \times 104$, where $0 \leq i < n$
- It is guaranteed that a solution always exists.

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function. The first line contains an integer n, the size of the array arr. Each of the next n lines contains an integer, arr[i], where $0 \leq i < n$.

Sample Input

STDIN		Function Parameters
4	→	arr[] size n = 4
1	→	arr = [1, 2, 3, 3]
2		
3		
3		

Sample Output 0
2

Explanation 0

- The sum of the first two elements, 1+2=3. The value of the last element is 3.
- Using zero based indexing, arr[2]=3 is the pivot between the two subarrays.
- The index of the pivot is 2.

Program:

```
int equalSum(int arr_count, int* arr)
{
    int t = 0, ls = 0;
    for (int i=0; i<arr_count; i++){
        t += arr[i];
    }
    for (int i=0; i<arr_count; i++){
        t -= arr[i];
        if (ls == t){
            return i;
        }
        ls += arr[i];
    }
    return -1;
}
```

Ex. No. : Date :

Sum Them All

Problem Statement:

Calculate the sum of an array of integers.

Example
numbers = [3, 13, 4, 11, 9]
The sum is 3 + 13 + 4 + 11 + 9 = 40.

Function Description
Complete the function arraySum in the editor below.

arraySum has the following parameter(s):
int numbers[n]: an array of integers

Returns
int: integer sum of the numbers array

Constraints
$1 \leq n \leq 104$
$1 \leq$ numbers[i] ≤ 104

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.
The first line contains an integer n, the size of the array numbers.
Each of the next n lines contains an integer numbers[i] where $0 \leq i < n$.

Sample Input

STDIN		Function
5	→	numbers[] size n = 5
1	→	numbers = [1, 2, 3, 4, 5]
2		
3		
4		
5		

Sample Output
15

Explanation
1 + 2 + 3 + 4 + 5 = 15.

Program:

```
int arraysum(int numbers_count, i+){
        int sum = 0;
        for (int i = 0; i < numbers_count ; i++){
                sum + = numbers [i];
        }
        return sum;
}
```

Ex. No. : Date :

Minimum Difference Sum

Problem Statement:

Given an array of n integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences.

Example
n = 5, arr = [1, 3, 3, 3, 4]
If the list is rearranged as arr' = [1, 3, 3, 3, 4], the absolute differences are |1 - 2| = 1, |3 - 3| = 1, |3 - 3| = 0, |3 - 4| = 1. The sum of those differences is 1 + 1 + 0 + 1 = 3.

Function Description
Complete the function minDiff in the editor below.

minDiff has the following parameter:
arr: an integer array

Returns:
int: the sum of the absolute differences of adjacent elements

Constraints
$2 \leq n \leq 10^3$
$0 \leq arr[i] \leq 10^9$, where $0 \leq i < n$

Input Format For Custom Testing
The first line of input contains an integer, n, the size of arr.
Each of the following n lines contains an integer that describes arr[i] (where $0 \leq i < n$) .

Sample Input For Custom Testing

STDIN		Function
5	→	arr[] size n = 5
5	→	arr[] = [5, 1, 3, 7, 3]
1		
3		
7		
3		

Sample Output
6

Explanation
n = 5, arr = [5, 1, 3, 7, 3]
If arr is rearranged as arr' = [1, 3, 3, 5, 7], the differences are minimized.
The final answer is |1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6.

Program:

```
int compare(const void *a, const void *b){
    return (*(int*)a - *(int*)b);
}
int minDiff(int arr[], int* arr)
{
    qsort(arr, arr_count, sizeof(int), compare);
    int min_sum = 0;
    for(int i = 1; i<arr_count; i++){
        min_sum += abs(arr[i]-arr[i-1]);
    }
    return min_sum;
}
```

Result:

Thus, the program is implemented and executed successfully

Structures and Unions

Ex. No. 1 Date :

Boxes through a Tunnel

Problem Statement:

You are transporting some boxes through a tunnel, where each box is a parallelepiped, and is characterized by its length, width and height.
The height of the tunnel 41 feet and the width can be assumed to be infinite. A box can be carried through the tunnel only if its height is strictly less than the tunnel's height. Find the volume of each box that can be successfully transported to the other end of the tunnel.

Note: Boxes cannot be rotated.

Input Format
The first line contains a single integer n, denoting the number of boxes.
n lines follow with three integers on each separated by single spaces - $length_i$, $width_i$ and $height_i$ which are length, width and height in feet of the i-th box.

Constraints

$1 \leq n \leq 100$

$1 \leq length_i, width_i, height_i \leq 100$

Output Format
For every box from the input which has a height lesser than 41 feet, print its volume in a separate line.

Sample Input
4
5 5 5
1 2 40
10 5 41
7 2 42

Sample Output
125
80

Explanation
The first box is really low, only 5 feet tall, so it can pass through the tunnel and its volume is 5 x 5 x 5 = 125.

The second box is sufficiently low, its volume is 1 x 2 x 4= = 80.

The third box is exactly 41 feet tall, so it cannot pass. The same can be said about the fourth box.

Program:

```
#include <stdio.h>
int main(){
    int n,i;
    int length,width,height;
    scanf("%d",&n);
    for(i=0;i<n;i++){
        scanf("%d %d %d",&length,&width,&height);
        if(height<41){
            printf("%d\n",length*width*height);
        }
    }
    return 0;
}
```

Ex. No. : Date :

Small Triangles, Large Triangles

Problem Statement:

You are given n triangles, specifically, their sides a_i, b_i and c_i. Print them in the same style but sorted by their areas from the smallest one to the largest one. It is guaranteed that all the areas are different.

The best way to calculate a volume of the triangle with sides a, b and c is Heron's formula:

$S = \sqrt{p * (p - a) * (p - b) * (p - c)}$ where $p = (a + b + c) / 2$.

Input Format
First line of each test file contains single integer n. n lines follow with ai, bi and ci on each separated by single spaces.

Constraints
$1 \leq n \leq 100$
$1 \leq a_i, b_i, c_i \leq 70$
$a_i + b_i > c_i$, $a_i + c_i > b_i$ and $b_i + c_i > a_i$

Output Format
Print exactly n lines. On each line print 3 integers separated by single spaces, which are ai, bi and ci of the corresponding triangle.

Sample Input
3
7 24 25
5 12 13
3 4 5

Sample Output
3 4 5
5 12 13
7 24 25

Explanation
The square of the first triangle is 84. The square of the second triangle is 30. The square of the third triangle is 6. So, the sorted order is the reverse one.

Program:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
int main(){
    int n;
    scanf("%d", &n);
    double a[n], b[n], c[n], area[n];
    for(int i=0;i<n;i++){
        scanf("%lf %lf %lf", &a[i], &b[i], &c[i]);
        double s=(a[i]+b[i]+c[i])/2.0;
        area[i]=sqrt(s*(s-a[i])*(s-b[i])*(s-c[i]));
    }
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(area[j]>area[j+1]){
                double temp_area;
                area[j]=area[j+1];
                area[j+1]=temp_area;

                double temp_a=a[j];
                double temp_b=b[j];
                double temp_c=c[j];
                a[j]=a[j+1];
                b[j]=b[j+1];
                c[j]=c[j+1];
                a[j+1]=temp_a;
                b[j+1]=temp_b;
```

```
                c[j+1] = temp;
            }
        }
    }
    for (int i=0; i<n; i++){
        printf("%.0f %.0f %.0f \n", a[i], b[i], c[i]);
    }
    return 0;
}
```

Result:

Thus, the c program is implemented and executed successfully

Pointers

Ex. No. :

Date :

Reverse a List

Problem Statement:

Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

Example
arr = [1, 3, 2, 4, 5]
Return the array [5, 4, 2, 3, 1] which is the reverse of the input array.

Function Description
Complete the function reverseArray in the editor below.

reverseArray has the following parameter(s):
int arr[n]: an array of integers

Return
int[n]: the array in reverse order

Constraints
$1 \leq n \leq 100$
$0 < arr[i] \leq 100$

Input Format For Custom Testing
The first line contains an integer, n, the number of elements in arr.
Each line i of the n subsequent lines (where $0 \leq i < n$) contains an integer, arr[i].

Sample Input For Custom Testing
5
1
3
2
4
5

Sample Output
5
4
2
3
1

Explanation
The input array is [1, 3, 2, 4, 5], so the reverse of the input array is [5, 4, 2, 3, 1].

Program:

```
int* reverseArray(int arr_count, int *arr, int *result_count)
{
    *result_count = arr_count;
    int* reversedarray = (int *)malloc(arr_count*sizeof
                                       (int));
    for (int i=0; i<arr_count; i++){
        reversedarray[i] = arr[arr_count-i-1];
    }
    return reversedarray;
}
```

Ex. No. :

Date :

Maximise the Value

Problem Statement:

Rearrange an array of integers so that the calculated value U is maximized. Among the arrangements that satisfy that test, choose the array with minimal ordering. The value of U for an array with n elements is calculated as:

U = arr[0]×arr[2]×(1÷arr[3])×arr[4]×...×arr[n-1] × (1÷arr[n]) if n is odd (or)
U = arr[0]×arr[2]×(1÷arr[3])×arr[4]×...×(1÷arr[n-1]) × arr[n] if n is even

The sequence of operations is the same in either case, but the length of the array, n, determines whether the calculation ends on arr[n] or (1÷arr[n]). Arrange the elements to maximize U and the items are in the numerically smallest possible order.

Example: arr = [3, 7, 9, 21, 34]
To maximize U and minimize the order, arrange the array as [9, 21, 5, 34, 7] so U = 9 × 21 × (1÷5) × 34 × (1÷7) = 183.6. The same U can be achieved using several other orders, e.g. [21, 9, 7, 34, 5] = 21 × 9 × (1÷7) × 34 × (1÷5) = 183.6, but they are not in the minimal order.

Function Description: Complete the function rearrange in the editor below.
rearrange has the following parameter(s): int arr[n]: an array of integers
Returns: int[n]: the elements of arr rearranged as described
Constraints: 1 ≤ n ≤ 105, 1 ≤ n ≤ 109

Input Format For Custom Testing: The first line contains an integer, n, the number of elements in arr. Each line i of the n subsequent lines (where 1 ≤ i ≤ n) contains an integer, arr[i].

Sample Input For Custom Testing

STDIN		Function
4	→	arr[] size n = 4
1	→	arr = [1, 2, 3, 4]
2		
3		
4		

Sample Output
2
3
1
4

Explanation
U = 2×3×(1÷1)×4 = 24. All other arrangements where U = 24 are numerically higher than this array, e.g. [2, 3, 1, 4] < [3, 4, 1, 2].

Program:

```
char * can_them_All (int lengths_count, long *lengths, long minlength)
{
    long sum = 0;
    for (int i = 0; i < lengths_count; i++) {
        sum += length[i];
    }
    for (int i = 0; i < lengths_count; i++) {
        if (sum < min length) {
            break;
        }
        sum -= lengths[i];
        if (sum >= min length) {
            static char result[] = "Possible";
            return result;
        }
    }
    static char result[] = "Impossible";
    return result;
}
```