**Q.No: 1**

***Locker Problem***

There is a school with 100 students, and correspondingly 100 lockers, all of which start off closed. The first student opens every locker. The second student closes every other locker, starting with the second (2, 4, 6 etc). The third student changes the state of every third locker starting with the third (3,6,9 etc). The fourth would change the status of lockers numbered 4,8,12 etc.,. That is, if the locker is open, it is closed, and if it is closed, it is opened. This continues until all 100 students have passed along the lockers. After the 100th student is done, which lockers are open and which are closed?

[Note: program should work for any number of students/lockers]

Sample Input 1

100

Sample Output 1

open = 10  
close = 90

**Solution:**

***#include<stdio.h>***

***int main()***

***{***

***long long int N , square , ind = 0 , count = 0;***

***scanf("%lld" , &N);***

***square = 1;***

***ind = 2;***

***while(square <= N)***

***{***

***count++;***

***square = ind \* ind;***

***ind++;***

***}***

***printf("open = %lld\nclose = %lld" , count , N-count);***

***return 0;***

***}***

**Test Case:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Input** | **Output** | **Difficulty** |
| **1** | **100** | **open = 10**  **close = 90** | **SAmple** |
| **2** | **1000** | **open = 31**  **close = 969** | **HArd** |
| **3** | **6456** | **open = 80**  **close = 6376** | **Hard** |
| **4** | **234** | **open = 15**  **close = 219** | **Medium** |
| **5** | **6576** | **open = 81**  **close = 6495** | **Hard** |
| **6** | **775757** | **open = 880**  **close = 774877** | **Hard** |
| **7** | **5765476** | **open = 2401**  **close = 5763075** | **Hard** |
| **8** | **1038757** | **open = 1019**  **close = 1037738** | **Hard** |
| **9** | **88** | **open = 9**  **close = 79** | **Easy** |
| **10** | **7565** | **open = 86**  **close = 7479** | **Medium** |
| **11** | **646474** | **open = 804**  **close = 645670** | **Medium** |

**Q.No: 2**

***Find the Path***

**Given an (m x n) matrix, write a program to traverse the cell and print the values present in the given path. Inclued necessary validation and proper error messages in case of given path is out of bounds.**

**5 x 5 matrix :**

**{1 2 3 4 5 } (row 1)**

**{6 7 8 9 0 } (row 2)**

**{1 2 3 4 5 } (row 3)**

**{6 7 8 9 0 } (row 4)**

**{1 2 3 4 5 } (row 5)**

**Path Notation : ‘’>” is going right, “v” going down, “<” is going left, “^” is going up.**

**Example Input 1 :**

**Start at (Row, Column): 1, 2**

**Path: >>> v**

**Output: 2 3 4 5 0**

**Example Input 2:**

**Start at (Row, Column): 2,3**

**Path: v > > v < < ^ > > v v**

**Output 8 3 4 5 0 9 8 3 4 5 0 5**

**Example Input 3:**

**Start at(row, Column): 1 , 4**

**Path: > v > >**

**Output: Invalid Path**

Input Format

**N M - matrix row and col**

**input for matrix**

**startrow startcol**

**path string**

Sample Input 1

**5 5  
1 2 3 4 5  
6 7 8 9 0  
1 2 3 4 5  
6 7 8 9 0  
1 2 3 4 5  
2 3  
v>>v<<^>>vv**

Sample Output 1

**8 3 4 5 0 9 8 3 4 5 0 5**

**Solution:**

**#include<stdio.h>**

**#include<malloc.h>**

**#define isBoundC(col) (col >= 0 && col < M)**

**#define isBoundR(row) (row >= 0 && row < N)**

**int main()**

**{**

**int \*path , count = 0;**

**int N , len , M , row , col , flag,startrow , startcol , ind;**

**char str[100];**

**scanf("%d%d" ,&N,&M);**

**int arr[N][M];**

**for(row = 0 ; row < N ; row++)**

**{**

**for(col = 0 ; col < M ; col++)**

**scanf("%d" , &arr[row][col]);**

**}**

**scanf("%d%d",&startrow ,&startcol);**

**scanf("%s", str);**

**for(len = 0 ; str[len] ; len++);**

**path = (int\*)calloc(len+1 , sizeof(int));**

**if(isBoundR(startrow-1) && isBoundC(startcol-1))**

**{**

**startrow--;**

**startcol--;**

**path[count++] = arr[startrow][startcol];**

**for(ind = 0 , flag = 0 ; flag !=1 && str[ind] ; ind++)**

**{**

**switch(str[ind])**

**{**

**case '>' : if(isBoundC(startcol+1))**

**path[count++] = arr[startrow][++startcol];**

**else flag = 1 ;**

**break;**

**case '<' : if(isBoundC(startcol-1))**

**path[count++] = arr[startrow][--startcol];**

**else flag = 1 ;**

**break;**

**case '^' : if(isBoundR(startrow-1))**

**path[count++] = arr[--startrow][startcol];**

**else flag = 1 ;**

**break;**

**case 'v' : if(isBoundR(startrow+1))**

**path[count++] = arr[++startrow][startcol];**

**else flag = 1 ;**

**break;**

**}**

**}**

**if(flag == 1 || count == 0)**

**printf("Invalid Path");**

**else**

**{**

**for(ind = 0 ; ind < count ; ind++)**

**printf("%d ", path[ind]);**

**}**

**}**

**return 0;**

**}**

**Test CAses:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Input** | **Output** | **Difficulty** |
| **1** | **5 5**  **1 2 3 4 5**  **6 7 8 9 0**  **1 2 3 4 5**  **6 7 8 9 0**  **1 2 3 4 5**  **2 3**  **v>>v<<^>>vv** | **8 3 4 5 0 9 8 3 4 5 0 5** | **Sample** |
| **2** | **3 4**  **1 2 3 4**  **5 6 7 8**  **1 2 3 0**  **1 1**  **>>vv<>** | **1 2 3 7 3 2 3** | **Hard** |
| **3** | **5 6**  **1 2 3 4 5 6**  **7 8 9 0 1 2**  **1 2 3 4 5 6**  **7 8 9 0 1 2**  **1 2 3 4 5 6**  **3 3**  **<<>>^^^^** | **Invalid Path** | **Hard** |
| **4** | **5 6**  **1 2 3 4 5 6**  **7 8 9 0 1 2**  **1 2 3 4 5 6**  **7 8 9 0 1 2**  **1 2 3 4 5 6**  **3 3**  **<<>>^>v** | **3 2 1 2 3 9 0 4** | **Hard** |
| **5** | **1 7**  **1 2 3 4 5 6 7**  **1 1**  **>>>>>** | **1 2 3 4 5 6** | **Easy** |
| **6** | **5 5**  **1 2 3 4 5**  **6 7 8 9 0**  **6 7 8 9 0**  **1 2 3 4 5**  **2 2 2 2 2**  **1 4**  **<<vv><><** | **4 3 2 7 7 8 7 8 7** | **Hard** |
| **7** | **5 5**  **1 2 3 4 5**  **6 7 8 9 0**  **6 7 8 9 0**  **1 2 3 4 5**  **2 2 2 2 2**  **1 4**  **<<<<<<<<<<** | **Invalid Path** | **Hard** |
| **8** | **10 10**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **2 3**  **>>>><<vv^^>>vv<><>vv^><^<<>>vv^** | **8 7 6 5 4 5 6 5 6 5 6 5 4 7 4 5 4 5 4 7 4 7 8 7 4 5 6 5 4 7 4 7** | **HArd** |
| **9** | **10 10**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **1 2 3 4 5 6 7 8 9 0**  **0 9 8 7 6 5 4 3 2 1**  **2 3**  **>>>><<vv^^>>vv<><>vv^><^<<>>vv^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^** | **Invalid Path** | **Hard** |
| **10** | **3 3**  **1 2 3**  **3 2 1**  **1 2 3**  **1 1**  **>>v<<v>>^^<><><><><><>** | **1 2 3 1 2 3 1 2 3 1 3 2 3 2 3 2 3 2 3 2 3 2 3** | **Hard** |

**Q.No: 3**

**Group anagram words**

**Given array of words, group the anagrams and print. Any word or phrase that exactly reproduces the letters in another order is an anagram. Arrive most efficient algorithm.**

**Examples :**

**Input: {tar,rat,banana,atr,nanaba}**

**Output: Anagrams:**

**rat atr tar**

**nanaba banana**

**Input: {abc, cde, xyz, dec}**

**Output: Anagrams:**

**cde dec**

**Others:**

**Abc**

**xyz**

Input Format

**N - no of words**

**get input words into array**

Sample Input 1

**5  
tar rat banana atr nanaba**

Sample Output 1

**rat atr tar  
nanaba banana**

**Solution:**

#include<stdio.h>

#include<malloc.h>

int strLen(char \*str)

{

int ind;

for(ind = 0 ; str[ind ] ; ind++);

return ind;

}

int main()

{

char str[100][100];

int N , i , j , set = 0 , ind , ind1 , len , len1 , sum;

int \*there;

scanf("%d" , &N);

int result[N] , count = 0;

for(ind = 0 ; ind < N ; ind++)

scanf("%s" , str[ind]);

for(ind = 0 ; ind < N ; ind++ , set = 0)

{

if(str[ind])

len = strLen(str[ind]);

else

continue;

for(ind1 = ind + 1 ; ind1 < N ; ind1++)

{

if(str[ind1])

len1 = strLen(str[ind1]);

else

continue;

if(len == len1)

{

there = (int\*)calloc(26 , sizeof(int));

for(i = 0 ; i < len ; i++)

there [ str[ind][i] - 97 ]++;

for(j = 0 ; j < len ; j++)

{

if(there[str[ind1][j] - 97])

there[str[ind1][j] - 97]--;

else

break;

}

for(i = 0 , sum = 0; i<26 ; sum+= there[i++]);

if(sum == 0 && str[ind1][0])

{

printf("%s " , str[ind1]);

str[ind1][0] = 0;

set =1;

}

}

}

if(set ==1 && str[ind][0])

{

printf("%s", str[ind]);

printf("\n");

}

else if(str[ind][0])

result[count++] = ind;

}

for(i = 0 ; i < count ; i++)

printf("%s\n" , str[result[i]]);

return 0;

}

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Input** | **Output** | **Difficulty** |
| **1** | **5**  **tar rat banana atr nanaba** | **rat atr tar**  **nanaba banana** | **Sample** |
| **2** | **6**  **abc cab abcd bac dcba hdjd** | **cab bac abc**  **dcba abcd**  **hdjd** | **Hard** |
| **3** | **10**  **abc cab dhfh fhfhf abcd dcba dcbaa aaa cadb bacd** | **cab abc**  **dcba cadb bacd abcd**  **dhfh**  **fhfhf**  **dcbaa**  **aaa** | **Hard** |
| **4** | **7**  **hjfdjf hfdjhfd jhfjfg hjfdf hjdfjdf hfd hjfd** | **hjfdjf**  **hfdjhfd**  **jhfjfg**  **hjfdf**  **hjdfjdf**  **hfd**  **hjfd** | **Hard** |
| **5** | **5**  **hai aih iah hello elloh** | **aih iah hai**  **elloh hello** | **Hard** |
| **6** | **4**  **abc fhf jdfjg jd** | **abc**  **fhf**  **jdfjg**  **jd** | **Easy** |
| **7** | **6**  **abcdef fedcab abcd dcba abc cab** | **fedcab abcdef**  **dcba abcd**  **cab abc** | **Hard** |
| **8** | **3**  **nwlrbbmqbhcdarzowkkyhiddqscdxrjmowfrxsjybldbefsarc qscdxrjmowfrxsjybldbefsarcnwlrbbmqbhcdarzowkkyhidd hjfjdfhj** | **qscdxrjmowfrxsjybldbefsarcnwlrbbmqbhcdarzowkkyhidd nwlrbbmqbhcdarzowkkyhiddqscdxrjmowfrxsjybldbefsarc**  **hjfjdfhj** | **Hard** |
| **9** | **5**  **nwlrbbmqbhcdarzowkkyhiddqscdxrjmowfrxsjybldbefsarcbynecdyggxxpklorellnmpapqfwkho wfrxsjybldbefsarcbynecdyggxxpklorellnmpapqfwkhonwlrbbmqbhcdarzowkkyhiddqscdxrjmo nwlrbbmqbhcdarzowkkyhiddqscdxrjmowfrxsjybldbefsarc ddqscdxrjmowfrxsjybldbefsarcnwlrbbmqbhcdarzowkkyhi jdfjhfjkg** | **wfrxsjybldbefsarcbynecdyggxxpklorellnmpapqfwkhonwlrbbmqbhcdarzowkkyhiddqscdxrjmo nwlrbbmqbhcdarzowkkyhiddqscdxrjmowfrxsjybldbefsarcbynecdyggxxpklorellnmpapqfwkho**  **ddqscdxrjmowfrxsjybldbefsarcnwlrbbmqbhcdarzowkkyhi nwlrbbmqbhcdarzowkkyhiddqscdxrjmowfrxsjybldbefsarc**  **jdfjhfjkg** | **Hard** |
| **10** | **7**  **hai hello hai hello hai hello fhf** | **hai hai hai**  **hello hello hello**  **fhf** | **Easy** |
| **11** | **5**  **jfjf jjfjg jgg abc cba** | **cba abc**  **jfjf**  **jjfjg**  **jgg** | **Easy** |

**Q.No: 4**

***Permutation of string***

**A permutation, also called an “arrangement number” or “order,” is a rearrangement of the elements of an ordered list S into a one-to-one correspondence with S itself. A string of length n has n! permutation.**

**Below are the permutations of string ABC.**

**ABC ACB BAC BCA CBA CAB**

Sample Input 1

**ABC**

Sample Output 1

**ABC ACB BAC BCA CBA CAB**

Sample Input 2

**1234**

Sample Output 2

**1234 1243 1324 1342 1432 1423 2134 2143 2314 2341 2**

**Solution:**

**#include<stdio.h>**

**void swap(char \*x, char \*y)**

**{**

**char temp;**

**temp = \*x;**

**\*x = \*y;**

**\*y = temp;**

**}**

**void permutation(char \*str, int l, int r)**

**{**

**int i;**

**if (l == r)**

**printf("%s ", str);**

**else**

**{**

**for (i = l; i <= r; i++)**

**{**

**swap((str+l), (str+i));**

**permutation(str, l+1, r);**

**swap((str+l), (str+i)); //backtrack**

**}**

**}**

**}**

**int main()**

**{**

**char str[100];**

**int len;**

**scanf("%s" , str);**

**for(len = 0 ; str[len] ; len++);**

**permutation(str, 0, len-1);**

**return 0;**

**}**

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **Input** | **Output** | **Difficulty** |
| **1** | **ABC** | **ABC ACB BAC BCA CBA CAB** | **Sample** |
| **2** | **1234** | **1234 1243 1324 1342 1432 1423 2134 2143 2314 2341 2431 2413 3214 3241 3124 3142 3412 3421 4231 4213 4321 4312 4132 4123** | **Sample** |
| **3** | **1234** | **1234 1243 1324 1342 1432 1423 2134 2143 2314 2341 2431 2413 3214 3241 3124 3142 3412 3421 4231 4213 4321 4312 4132 4123** | **Hard** |
| **4** | **haihel** | **haihel haihle haiehl haielh haileh hailhe hahiel hahile haheil haheli hahlei hahlie haehil haehli haeihl haeilh haelih haelhi halhei halhie halehi haleih halieh halihe hiahel hiahle hiaehl hiaelh hialeh hialhe hihael hihale hiheal hihela hihlea hihlae hiehal hiehla hieahl hiealh hielah hielha hilhea hilhae hileha hileah hilaeh hilahe hhiael hhiale hhieal hhiela hhilea hhilae hhaiel hhaile hhaeil hhaeli hhalei hhalie hheail hheali hheial hheila hhelia hhelai hhlaei hhlaie hhleai hhleia hhliea hhliae heihal heihla heiahl heialh heilah heilha hehial hehila hehail hehali hehlai hehlia heahil heahli heaihl heailh healih healhi helhai helhia helahi helaih heliah heliha hlihea hlihae hlieha hlieah hliaeh hliahe hlhiea hlhiae hlheia hlheai hlhaei hlhaie hlehia hlehai hleiha hleiah hleaih hleahi hlahei hlahie hlaehi hlaeih hlaieh hlaihe ahihel ahihle ahiehl ahielh ahileh ahilhe ahhiel ahhile ahheil ahheli ahhlei ahhlie ahehil ahehli aheihl aheilh ahelih ahelhi ahlhei ahlhie ahlehi ahleih ahlieh ahlihe aihhel aihhle aihehl aihelh aihleh aihlhe aihhel aihhle aihehl aihelh aihleh aihlhe aiehhl aiehlh aiehhl aiehlh aielhh aielhh ailheh ailhhe ailehh ailehh ailheh ailhhe ahihel ahihle ahiehl ahielh ahileh ahilhe ahhiel ahhile ahheil ahheli ahhlei ahhlie ahehil ahehli aheihl aheilh ahelih ahelhi ahlhei ahlhie ahlehi ahleih ahlieh ahlihe aeihhl aeihlh aeihhl aeihlh aeilhh aeilhh aehihl aehilh aehhil aehhli aehlhi aehlih aehhil aehhli aehihl aehilh aehlih aehlhi aelhhi aelhih aelhhi aelhih aelihh aelihh aliheh alihhe aliehh aliehh aliheh alihhe alhieh alhihe alheih alhehi alhhei alhhie alehih alehhi aleihh aleihh alehih alehhi alhhei alhhie alhehi alheih alhieh alhihe iahhel iahhle iahehl iahelh iahleh iahlhe iahhel iahhle iahehl iahelh iahleh iahlhe iaehhl iaehlh iaehhl iaehlh iaelhh iaelhh ialheh ialhhe ialehh ialehh ialheh ialhhe ihahel ihahle ihaehl ihaelh ihaleh ihalhe ihhael ihhale ihheal ihhela ihhlea ihhlae ihehal ihehla iheahl ihealh ihelah ihelha ihlhea ihlhae ihleha ihleah ihlaeh ihlahe ihhael ihhale ihheal ihhela ihhlea ihhlae ihahel ihahle ihaehl ihaelh ihaleh ihalhe iheahl ihealh ihehal ihehla ihelha ihelah ihlaeh ihlahe ihleah ihleha ihlhea ihlhae iehhal iehhla iehahl iehalh iehlah iehlha iehhal iehhla iehahl iehalh iehlah iehlha ieahhl ieahlh ieahhl ieahlh iealhh iealhh ielhah ielhha ielahh ielahh ielhah ielhha ilhhea ilhhae ilheha ilheah ilhaeh ilhahe ilhhea ilhhae ilheha ilheah ilhaeh ilhahe ilehha ilehah ilehha ilehah ileahh ileahh ilaheh ilahhe ilaehh ilaehh ilaheh ilahhe haihel haihle haiehl haielh haileh hailhe hahiel hahile haheil haheli hahlei hahlie haehil haehli haeihl haeilh haelih haelhi halhei halhie halehi haleih halieh halihe hiahel hiahle hiaehl hiaelh hialeh hialhe hihael hihale hiheal hihela hihlea hihlae hiehal hiehla hieahl hiealh hielah hielha hilhea hilhae hileha hileah hilaeh hilahe hhiael hhiale hhieal hhiela hhilea hhilae hhaiel hhaile hhaeil hhaeli hhalei hhalie hheail hheali hheial hheila hhelia hhelai hhlaei hhlaie hhleai hhleia hhliea hhliae heihal heihla heiahl heialh heilah heilha hehial hehila hehail hehali hehlai hehlia heahil heahli heaihl heailh healih healhi helhai helhia helahi helaih heliah heliha hlihea hlihae hlieha hlieah hliaeh hliahe hlhiea hlhiae hlheia hlheai hlhaei hlhaie hlehia hlehai hleiha hleiah hleaih hleahi hlahei hlahie hlaehi hlaeih hlaieh hlaihe eaihhl eaihlh eaihhl eaihlh eailhh eailhh eahihl eahilh eahhil eahhli eahlhi eahlih eahhil eahhli eahihl eahilh eahlih eahlhi ealhhi ealhih ealhhi ealhih ealihh ealihh eiahhl eiahlh eiahhl eiahlh eialhh eialhh eihahl eihalh eihhal eihhla eihlha eihlah eihhal eihhla eihahl eihalh eihlah eihlha eilhha eilhah eilhha eilhah eilahh eilahh ehiahl ehialh ehihal ehihla ehilha ehilah ehaihl ehailh ehahil ehahli ehalhi ehalih ehhail ehhali ehhial ehhila ehhlia ehhlai ehlahi ehlaih ehlhai ehlhia ehliha ehliah ehihal ehihla ehiahl ehialh ehilah ehilha ehhial ehhila ehhail ehhali ehhlai ehhlia ehahil ehahli ehaihl ehailh ehalih ehalhi ehlhai ehlhia ehlahi ehlaih ehliah ehliha elihha elihah elihha elihah eliahh eliahh elhiha elhiah elhhia elhhai elhahi elhaih elhhia elhhai elhiha elhiah elhaih elhahi elahhi elahih elahhi elahih elaihh elaihh laiheh laihhe laiehh laiehh laiheh laihhe lahieh lahihe laheih lahehi lahhei lahhie laehih laehhi laeihh laeihh laehih laehhi lahhei lahhie lahehi laheih lahieh lahihe liaheh liahhe liaehh liaehh liaheh liahhe lihaeh lihahe liheah liheha lihhea lihhae liehah liehha lieahh lieahh liehah liehha lihhea lihhae liheha liheah lihaeh lihahe lhiaeh lhiahe lhieah lhieha lhihea lhihae lhaieh lhaihe lhaeih lhaehi lhahei lhahie lheaih lheahi lheiah lheiha lhehia lhehai lhhaei lhhaie lhheai lhheia lhhiea lhhiae leihah leihha leiahh leiahh leihah leihha lehiah lehiha lehaih lehahi lehhai lehhia leahih leahhi leaihh leaihh leahih leahhi lehhai lehhia lehahi lehaih lehiah lehiha lhihea lhihae lhieha lhieah lhiaeh lhiahe lhhiea lhhiae lhheia lhheai lhhaei lhhaie lhehia lhehai lheiha lheiah lheaih lheahi lhahei lhahie lhaehi lhaeih lhaieh lhaihe** | **Hard** |
| **5** | **wonder** | **wonder wondre wonedr wonerd wonred wonrde wodner wodnre wodenr wodern wodren wodrne woednr woedrn woendr woenrd woernd woerdn worden wordne woredn worend worned wornde wnoder wnodre wnoedr wnoerd wnored wnorde wndoer wndore wndeor wndero wndreo wndroe wnedor wnedro wneodr wneord wnerod wnerdo wnrdeo wnrdoe wnredo wnreod wnroed wnrode wdnoer wdnore wdneor wdnero wdnreo wdnroe wdoner wdonre wdoenr wdoern wdoren wdorne wdeonr wdeorn wdenor wdenro wderno wderon wdroen wdrone wdreon wdreno wdrneo wdrnoe wendor wendro wenodr wenord wenrod wenrdo wednor wednro wedonr wedorn wedron wedrno weodnr weodrn weondr weonrd weornd weordn werdon werdno werodn werond wernod werndo wrndeo wrndoe wrnedo wrneod wrnoed wrnode wrdneo wrdnoe wrdeno wrdeon wrdoen wrdone wredno wredon wrendo wrenod wreond wreodn wroden wrodne wroedn wroend wroned wronde ownder owndre ownedr ownerd ownred ownrde owdner owdnre owdenr owdern owdren owdrne owednr owedrn owendr owenrd owernd owerdn owrden owrdne owredn owrend owrned owrnde onwder onwdre onwedr onwerd onwred onwrde ondwer ondwre ondewr onderw ondrew ondrwe onedwr onedrw onewdr onewrd onerwd onerdw onrdew onrdwe onredw onrewd onrwed onrwde odnwer odnwre odnewr odnerw odnrew odnrwe odwner odwnre odwenr odwern odwren odwrne odewnr odewrn odenwr odenrw odernw oderwn odrwen odrwne odrewn odrenw odrnew odrnwe oendwr oendrw oenwdr oenwrd oenrwd oenrdw oednwr oednrw oedwnr oedwrn oedrwn oedrnw oewdnr oewdrn oewndr oewnrd oewrnd oewrdn oerdwn oerdnw oerwdn oerwnd oernwd oerndw orndew orndwe ornedw ornewd ornwed ornwde ordnew ordnwe ordenw ordewn ordwen ordwne orednw oredwn orendw orenwd orewnd orewdn orwden orwdne orwedn orwend orwned orwnde nowder nowdre nowedr nowerd nowred nowrde nodwer nodwre nodewr noderw nodrew nodrwe noedwr noedrw noewdr noewrd noerwd noerdw nordew nordwe noredw norewd norwed norwde nwoder nwodre nwoedr nwoerd nwored nworde nwdoer nwdore nwdeor nwdero nwdreo nwdroe nwedor nwedro nweodr nweord nwerod nwerdo nwrdeo nwrdoe nwredo nwreod nwroed nwrode ndwoer ndwore ndweor ndwero ndwreo ndwroe ndower ndowre ndoewr ndoerw ndorew ndorwe ndeowr ndeorw ndewor ndewro nderwo nderow ndroew ndrowe ndreow ndrewo ndrweo ndrwoe newdor newdro newodr neword newrod newrdo nedwor nedwro nedowr nedorw nedrow nedrwo neodwr neodrw neowdr neowrd neorwd neordw nerdow nerdwo nerodw nerowd nerwod nerwdo nrwdeo nrwdoe nrwedo nrweod nrwoed nrwode nrdweo nrdwoe nrdewo nrdeow nrdoew nrdowe nredwo nredow nrewdo nrewod nreowd nreodw nrodew nrodwe nroedw nroewd nrowed nrowde donwer donwre donewr donerw donrew donrwe downer downre dowenr dowern dowren dowrne doewnr doewrn doenwr doenrw doernw doerwn dorwen dorwne dorewn dorenw dornew dornwe dnower dnowre dnoewr dnoerw dnorew dnorwe dnwoer dnwore dnweor dnwero dnwreo dnwroe dnewor dnewro dneowr dneorw dnerow dnerwo dnrweo dnrwoe dnrewo dnreow dnroew dnrowe dwnoer dwnore dwneor dwnero dwnreo dwnroe dwoner dwonre dwoenr dwoern dworen dworne dweonr dweorn dwenor dwenro dwerno dweron dwroen dwrone dwreon dwreno dwrneo dwrnoe denwor denwro denowr denorw denrow denrwo dewnor dewnro dewonr deworn dewron dewrno deownr deowrn deonwr deonrw deornw deorwn derwon derwno derown deronw dernow dernwo drnweo drnwoe drnewo drneow drnoew drnowe drwneo drwnoe drweno drweon drwoen drwone drewno drewon drenwo drenow dreonw dreown drowen drowne droewn droenw dronew dronwe eondwr eondrw eonwdr eonwrd eonrwd eonrdw eodnwr eodnrw eodwnr eodwrn eodrwn eodrnw eowdnr eowdrn eowndr eownrd eowrnd eowrdn eordwn eordnw eorwdn eorwnd eornwd eorndw enodwr enodrw enowdr enowrd enorwd enordw endowr endorw endwor endwro endrwo endrow enwdor enwdro enwodr enword enwrod enwrdo enrdwo enrdow enrwdo enrwod enrowd enrodw ednowr ednorw ednwor ednwro ednrwo ednrow edonwr edonrw edownr edowrn edorwn edornw edwonr edworn edwnor edwnro edwrno edwron edrown edronw edrwon edrwno edrnwo edrnow ewndor ewndro ewnodr ewnord ewnrod ewnrdo ewdnor ewdnro ewdonr ewdorn ewdron ewdrno ewodnr ewodrn ewondr ewonrd ewornd ewordn ewrdon ewrdno ewrodn ewrond ewrnod ewrndo erndwo erndow ernwdo ernwod ernowd ernodw erdnwo erdnow erdwno erdwon erdown erdonw erwdno erwdon erwndo erwnod erwond erwodn erodwn erodnw erowdn erownd eronwd erondw rondew rondwe ronedw ronewd ronwed ronwde rodnew rodnwe rodenw rodewn rodwen rodwne roednw roedwn roendw roenwd roewnd roewdn rowden rowdne rowedn rowend rowned rownde rnodew rnodwe rnoedw rnoewd rnowed rnowde rndoew rndowe rndeow rndewo rndweo rndwoe rnedow rnedwo rneodw rneowd rnewod rnewdo rnwdeo rnwdoe rnwedo rnweod rnwoed rnwode rdnoew rdnowe rdneow rdnewo rdnweo rdnwoe rdonew rdonwe rdoenw rdoewn rdowen rdowne rdeonw rdeown rdenow rdenwo rdewno rdewon rdwoen rdwone rdweon rdweno rdwneo rdwnoe rendow rendwo renodw renowd renwod renwdo rednow rednwo redonw redown redwon redwno reodnw reodwn reondw reonwd reownd reowdn rewdon rewdno rewodn rewond rewnod rewndo rwndeo rwndoe rwnedo rwneod rwnoed rwnode rwdneo rwdnoe rwdeno rwdeon rwdoen rwdone rwedno rwedon rwendo rwenod rweond rweodn rwoden rwodne rwoedn rwoend rwoned rwonde** | **Hard** |
| **6** | **12345** | **12345 12354 12435 12453 12543 12534 13245 13254 13425 13452 13542 13524 14325 14352 14235 14253 14523 14532 15342 15324 15432 15423 15243 15234 21345 21354 21435 21453 21543 21534 23145 23154 23415 23451 23541 23514 24315 24351 24135 24153 24513 24531 25341 25314 25431 25413 25143 25134 32145 32154 32415 32451 32541 32514 31245 31254 31425 31452 31542 31524 34125 34152 34215 34251 34521 34512 35142 35124 35412 35421 35241 35214 42315 42351 42135 42153 42513 42531 43215 43251 43125 43152 43512 43521 41325 41352 41235 41253 41523 41532 45312 45321 45132 45123 45213 45231 52341 52314 52431 52413 52143 52134 53241 53214 53421 53412 53142 53124 54321 54312 54231 54213 54123 54132 51342 51324 51432 51423 51243 51234** | **Hard** |
| **7** | **657547** | **657547 657574 657457 657475 657745 657754 655747 655774 655477 655477 655747 655774 654577 654577 654757 654775 654775 654757 657547 657574 657457 657475 657745 657754 675547 675574 675457 675475 675745 675754 675547 675574 675457 675475 675745 675754 674557 674575 674557 674575 674755 674755 677545 677554 677455 677455 677545 677554 657547 657574 657457 657475 657745 657754 655747 655774 655477 655477 655747 655774 654577 654577 654757 654775 654775 654757 657547 657574 657457 657475 657745 657754 647557 647575 647557 647575 647755 647755 645757 645775 645577 645577 645757 645775 645577 645577 645757 645775 645775 645757 647557 647575 647557 647575 647755 647755 677545 677554 677455 677455 677545 677554 675745 675754 675475 675457 675547 675574 674575 674557 674755 674755 674575 674557 675547 675574 675457 675475 675745 675754 567547 567574 567457 567475 567745 567754 565747 565774 565477 565477 565747 565774 564577 564577 564757 564775 564775 564757 567547 567574 567457 567475 567745 567754 576547 576574 576457 576475 576745 576754 575647 575674 575467 575476 575746 575764 574567 574576 574657 574675 574765 574756 577546 577564 577456 577465 577645 577654 557647 557674 557467 557476 557746 557764 556747 556774 556477 556477 556747 556774 554677 554677 554767 554776 554776 554767 557647 557674 557467 557476 557746 557764 547567 547576 547657 547675 547765 547756 545767 545776 545677 545677 545767 545776 546577 546577 546757 546775 546775 546757 547567 547576 547657 547675 547765 547756 577546 577564 577456 577465 577645 577654 575746 575764 575476 575467 575647 575674 574576 574567 574756 574765 574675 574657 576547 576574 576457 576475 576745 576754 756547 756574 756457 756475 756745 756754 755647 755674 755467 755476 755746 755764 754567 754576 754657 754675 754765 754756 757546 757564 757456 757465 757645 757654 765547 765574 765457 765475 765745 765754 765547 765574 765457 765475 765745 765754 764557 764575 764557 764575 764755 764755 767545 767554 767455 767455 767545 767554 756547 756574 756457 756475 756745 756754 755647 755674 755467 755476 755746 755764 754567 754576 754657 754675 754765 754756 757546 757564 757456 757465 757645 757654 746557 746575 746557 746575 746755 746755 745657 745675 745567 745576 745756 745765 745567 745576 745657 745675 745765 745756 747556 747565 747556 747565 747655 747655 776545 776554 776455 776455 776545 776554 775645 775654 775465 775456 775546 775564 774565 774556 774655 774655 774565 774556 775546 775564 775456 775465 775645 775654 557647 557674 557467 557476 557746 557764 556747 556774 556477 556477 556747 556774 554677 554677 554767 554776 554776 554767 557647 557674 557467 557476 557746 557764 575647 575674 575467 575476 575746 575764 576547 576574 576457 576475 576745 576754 574657 574675 574567 574576 574756 574765 577645 577654 577465 577456 577546 577564 567547 567574 567457 567475 567745 567754 565747 565774 565477 565477 565747 565774 564577 564577 564757 564775 564775 564757 567547 567574 567457 567475 567745 567754 547657 547675 547567 547576 547756 547765 546757 546775 546577 546577 546757 546775 545677 545677 545767 545776 545776 545767 547657 547675 547567 547576 547756 547765 577645 577654 577465 577456 577546 577564 576745 576754 576475 576457 576547 576574 574675 574657 574765 574756 574576 574567 575647 575674 575467 575476 575746 575764 457567 457576 457657 457675 457765 457756 455767 455776 455677 455677 455767 455776 456577 456577 456757 456775 456775 456757 457567 457576 457657 457675 457765 457756 475567 475576 475657 475675 475765 475756 475567 475576 475657 475675 475765 475756 476557 476575 476557 476575 476755 476755 477565 477556 477655 477655 477565 477556 457567 457576 457657 457675 457765 457756 455767 455776 455677 455677 455767 455776 456577 456577 456757 456775 456775 456757 457567 457576 457657 457675 457765 457756 467557 467575 467557 467575 467755 467755 465757 465775 465577 465577 465757 465775 465577 465577 465757 465775 465775 465757 467557 467575 467557 467575 467755 467755 477565 477556 477655 477655 477565 477556 475765 475756 475675 475657 475567 475576 476575 476557 476755 476755 476575 476557 475567 475576 475657 475675 475765 475756 757546 757564 757456 757465 757645 757654 755746 755764 755476 755467 755647 755674 754576 754567 754756 754765 754675 754657 756547 756574 756457 756475 756745 756754 775546 775564 775456 775465 775645 775654 775546 775564 775456 775465 775645 775654 774556 774565 774556 774565 774655 774655 776545 776554 776455 776455 776545 776554 757546 757564 757456 757465 757645 757654 755746 755764 755476 755467 755647 755674 754576 754567 754756 754765 754675 754657 756547 756574 756457 756475 756745 756754 747556 747565 747556 747565 747655 747655 745756 745765 745576 745567 745657 745675 745576 745567 745756 745765 745675 745657 746557 746575 746557 746575 746755 746755 767545 767554 767455 767455 767545 767554 765745 765754 765475 765457 765547 765574 764575 764557 764755 764755 764575 764557 765547 765574 765457 765475 765745 765754** | **Hard** |
| **8** | **123** | **123 132 213 231 321 312** | **Easy** |
| **9** | **gfhg** | **gfhg gfgh ghfg ghgf gghf ggfh fghg fggh fhgg fhgg fghg fggh hfgg hfgg hgfg hggf hggf hgfg gfhg gfgh ghfg ghgf gghf ggfh** | **Medium** |
| **10** | **!#$%** | **!#$% !#%$ !$#% !$%# !%$# !%#$ #!$% #!%$ #$!% #$%! #%$! #%!$ $#!% $#%! $!#% $!%# $%!# $%#! %#$! %#!$ %$#! %$!# %!$# %!#$** | **Easy** |
| **11** | **<>^V** | **<>^V <>V^ <^>V <^V> <V^> <V>^ ><^V ><V^ >^<V >^V< >V^< >V<^ ^><V ^>V< ^<>V ^<V> ^V<> ^V>< V>^< V><^ V^>< V^<> V<^> V<>^** | **Hard** |

**Q.No: 5**

***Word Reversal form the first occurrence of sub-string***

**Write a program to accept two strings S1 and S2 and reverse the words of S1,starting from the word where the first occurrence of S2 present in S1. Same empty spaces between the words must be maintained in the output. Write the program without splitting up the strings into array of words.**

**DON'T use any inbuilt functions**

**Input: S1= This is a test input string S2=st**

**Output : This is a string input test**

Sample Input 1

**this is a test sentence  
st**

Sample Output 1

**this is a sentence test**

**Solution:**

**#include<stdio.h>**

**char \* strReverse(char \*str)**

**{**

**int start , end ;**

**char temp;**

**for(end = 0 ; str[end] ; end++);**

**for(start = 0 , end-- ; start < end ; start++ , end--)**

**{**

**temp = str[start];**

**str[start] = str[end];**

**str[end] = temp;**

**}**

**return str;**

**}**

**int substring(char \*s1 , char \*s2)**

**{**

**int ind1 , ind2 , start = 0 , i , j;**

**for(ind1 = 0 ; s1[ind1] ; ind1++)**

**{**

**if(s1[ind1] == 32)**

**start = ind1 + 1;**

**else if(s1[ind1] == s2[0])**

**{**

**for(i= ind1+1 , j = 1 ; s1[i] && s2[j] && s1[i] == s2[j] ; i++ , j++);**

**if(s2[j] == 0)**

**return start;**

**}**

**}**

**return -1;**

**}**

**int main()**

**{**

**char str1[1000] , str2 [1000];**

**char \*start;**

**int ind, st;**

**scanf("%[^\n]s" , str1);**

**scanf("%s" , str2);**

**st = substring(str1 , str2);**

**start = str1 + st ;**

**strReverse(start);**

**for(ind = st ; str1[ind] ; ind++)**

**{**

**if(str1[ind] == 32)**

**{**

**str1[ind] = 0;**

**strReverse(start);**

**start = str1 + ind + 1;**

**str1[ind] = 32;**

**}**

**}**

**strReverse(start);**

**printf("%s" , str1);**

**}**

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Count** | **input** | **Output** | **Difficulty** |
| **1** | **this is a test sentence**  **st** | **this is a sentence test** | **Sample** |
| **2** | **one two three four five**  **ree** | **one two five four three** | **Hard** |
| **3** | **idjbc jdsnc ookjdncv iokndfvkjn iokndcvkln ionfvjnjknv jkndfvjknv jknfvjkfnvkjf vkjnfvjknfk vjnfvjkndfkv kjnvkjdfnkv jknvkjn**  **dncv** | **idjbc jdsnc kjnvkjddncv vjnfvjkndfkv vkjnfvjknfk jknfvjkfnvkjf jkndfvjknv ionfvjnjknv iokndcvkln iokndfvkjn ookjdncv** | **Hard** |
| **4** | **jdgchjdgsjvhgdvjhdfvj hjcvhjdvjhcvdhcvjvc dhvchjdvchvdchjvdjch hdcvhjdvchjdvc hdvchjdvcvdc hdvchjvdcvdchdvchj hvdvdc**  **cvhjdvchjd** | **jdgchjdgsjvhgdvjhdfvj hjcvhjdvjhcvdhcvjvc dhvchjdvchvdchjvdjch hvcvhjdvchjd hdvchjvdcvdchdvchj hdvchjdvcvdc hdcvhjdvchjdvc** | **Hard** |
| **5** | **hai hello how are you hai hello**  **ello** | **hai hello hai you are how hello** | **Hard** |
| **6** | **wonder wonderlaaa wonderfull**  **wonderful** | **wonder wonderlaaa wonderfull** | **Hard** |
| **7** | **wonder wonderlaaa wonderfull hjdsjcfhsdjhcvdcvhjdvchjdvsjc hjdbchjdvc hjvdjchdvc hdbcjhdvcj djhdvc hdvchjdvjch hjdvchjvdc hvchjdvcj hjdsjcfhs djhcvd cvhjdvchjdvsjc hjdbchjdvc hjvdjchdvc hdbcjhdvcj djhdvc hdvchjdvjch hjdvchjvdc hvchjdvcj**  **wonderful** | **wonder wonderlaaa hvchjdvcj hjdvchjvdc hdvchjdvjch djhdvc hdbcjhdvcj hjvdjchdvc hjdbchjdvc cvhjdvchjdvsjc djhcvd hjdsjcfhs hvchjdvcj hjdvchjvdc hdvchjdvjch djhdvc hdbcjhdvcj hjvdjchdvc hjdbchjdvc hjdsjcfhsdjhcvdcvhjdvchjdvsjc wonderfull** | **Hard** |
| **8** | **one one one one two three four**  **ne** | **four three two one one one one** | **Hard** |
| **9** | **djhvdjch hdbhjv dhjdvc dhcvjhdvcjdhvcjhvdj hjdbvjhdvjdv hjdbhjdvjhdvc hdbjhdbvjdv hjdbvhjdvhjd jhdvjhdvhjdvj dhvchjdvchjdvc hjdchjdvjchv hjdsjvc**  **dvc** | **djhvdjch hdbhjv hjdsjvc hjdchjdvjchv dhvchjdvchjdvc jhdvjhdvhjdvj hjdbvhjdvhjd hdbjhdbvjdv hjdbhjdvjhdvc hjdbvjhdvjdv dhcvjhdvcjdhvcjhvdj dhjdvc** | **Hard** |
| **10** | **i love india**  **ve** | **i india love** | **Easy** |
| **11** | **hai haiii haiiii**  **aiii** | **hai haiiii haiii** | **Easy** |