**Date : 24/04/2023**

**1. Prog for palindrom string or not ?**

#include <stdio.h>

#include <string.h>

int main() {

char str[20];

int i, len, is\_palindrome = 1;

printf("Enter a string: ");

scanf("%s", str);

len = strlen(str);

for(i=0; i<len/2; i++) {

if(str[i] != str[len-1-i]) {

is\_palindrome = 0;

break;

}

}

if(is\_palindrome) {

printf("%s is a palindrome.\n", str);

}

else {

printf("%s is not a palindrome.\n", str);

}

return 0;

}

**2. Write a prog for Dutch National Flag prog in c ?**

#include <stdio.h>

int main() {

int l[] = {0,0,1,2,1,1,2,0};

int low = 0, mid = 0, high = sizeof(l) / sizeof(int) - 1;

while (mid <= high) {

if (l[mid] == 1) {

mid++;

} else if (l[mid] == 0) {

int temp = l[low];

l[low] = l[mid];

l[mid] = temp;

low++;

mid++;

} else {

int temp = l[high];

l[high] = l[mid];

l[mid] = temp;

high--;

}

}

for (int i = 0; i < sizeof(l) / sizeof(int); i++) {

printf("%d ", l[i]);

}

printf("\n");

return 0;

}

**3. Check wether the given string’s are angram string or not write prog in c ?**

#include <stdio.h>

#include <string.h>

int main() {

char str1[100], str2[100];

int freq1[256] = {0}, freq2[256] = {0};

int i, len1, len2;

printf("Enter the first string: ");

scanf("%s", str1);

printf("Enter the second string: ");

scanf("%s", str2);

len1 = strlen(str1);

len2 = strlen(str2);

if (len1 != len2) {

printf("Strings are not anagrams.\n");

return 0;

}

for (i = 0; i < len1; i++) {

freq1[str1[i]]++;

freq2[str2[i]]++;

}

for (i = 0; i < 256; i++) {

if (freq1[i] != freq2[i]) {

printf("Strings are not anagrams.\n");

return 0;

}

}

printf("Strings are anagrams..........\n");

printf("Frequency of characters in string:\n");

for (i = 0; i < 256; i++) {

if (freq2[i] > 0) {

printf("%c: %d\n", i, freq2[i]);

}

}

return 0;

}

**Date : 25/04/2023**

**1. Write a prog for symmetric and palindrom ?**

#include <stdio.h>

#include <string.h>

int isSymmetric(char[], int);

int isPalindrome(char[], int);

int isPalindrome(char str[], int len)

{

for (int i = 0; i < len-1; i++)

{

if (str[i] != str[len - i - 1])

{

return 0; // not a palindrome

}

}

return 1; // is a palindrome

}

int isSymmetric(char str[], int len)

{

int mid;

if (len % 2 == 0) {

mid = len / 2;

} else {

mid = len / 2 + 1;

}

for (int i = 0; i < mid; i++)

{

if (str[i] != str[mid+i])

{

return 0; // not symmetric

}

}

return 1; // symmetric

}

int main()

{

char str[200];

printf("Enter any string :\n");

gets(str);

int len = strlen(str);

int isPalindromeFlag = isPalindrome(str, len);

int isSymmetricFlag = isSymmetric(str, len);

if (isPalindromeFlag == 1 && isSymmetricFlag==1)

printf("Given string is symmetric and palindrome\n");

else if (isPalindromeFlag == 0 && isSymmetricFlag==1)

printf("Given string is symmetric\n");

else if (isPalindromeFlag == 1 && isSymmetricFlag==0)

printf("Given string is palindrome\n");

else

printf("Given string is neither symmetric nor palindrome\n");

return 0;

}

**2. Write a prog for reverse the word in string ?**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main() {

char str[100];

int i, len;

printf("Enter a string:\n");

gets(str);

len = strlen(str);

printf("Reverse string order:\n");

for(i = len - 1; i >= 0; i--) {

if(str[i] == ' ' || i == 0) { // added condition to check for the last word

int j;

if(i == 0)

j = i;

else

j = i + 1;

while(str[j] != '\0' && str[j] != 32)

{

printf("%c", str[j]);

j++;

}

if (i > 0) // check if there are more words left to print

printf(" ");

}

}

return 0;

**Logical which is given before 24/04/2023**

**1.Reverse the vowel in the string like “helloa” then output become an “halloe” like write a prog in c ?**

#include <stdio.h>

#include <string.h>

int is\_vowel(char c) {

return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||

c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');

}

void reverse\_vowels(char\* str) {

int len = strlen(str);

int i = 0, j = len - 1;

while (i < j) {

if (!is\_vowel(str[i])) {

i++;

continue;

}

if (!is\_vowel(str[j])) {

j--;

continue;

}

// swap the vowels

char temp = str[i];

str[i] = str[j];

str[j] = temp;

i++;

j--;

}

}

int main() {

char str[100];

printf("Enter a string: ");

fgets(str, 100, stdin);

reverse\_vowels(str);

printf("Reversed vowels: %s", str);

return 0;

}

**2. 5 stairs there a person climbe 1 or 2 step at a time so no of way to climbe top write a prog in c**

#include <stdio.h>

// Recursive function to calculate the number of ways to climb stairs

int countWays(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return countWays(n-1) + countWays(n-2);

}

}

int main() {

int stairs = 5;

int ways = countWays(stairs);

printf("The number of ways to climb %d stairs is %d\n", stairs, ways);

return 0;

}

**3. Take array and array element is 1,2,4,3,5 and rotate the array element k time if k=2 them 2 time rotate the array element write a program in c**

#include <stdio.h>

void rotateArray(int arr[], int n, int k) {

k %= n; // reduce the number of rotations to be performed to k % n

while (k--) {

int temp = arr[0];

for (int i = 0; i < n - 1; i++) {

arr[i] = arr[i + 1];

}

arr[n - 1] = temp;

}

}

int main() {

int arr[] = {1, 2, 4, 3, 5};

int n = sizeof(arr) / sizeof(arr[0]);

int k = -3; // testing with negative k

printf("Array before rotation: ");

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

printf("%d ", arr[i]);

}

rotateArray(arr, n, k);

printf("\nArray after rotation: ");

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

printf("%d ", arr[i]);

}

return 0;

}

**Date : 28/04/2023 Friday**

**Sort an list element with less timecomplexity ?**

#include <stdio.h>

void quicksort(int arr[], int left, int right) {

int i = left, j = right, pivot = arr[(left + right) / 2];

while (i <= j) {

while (arr[i] < pivot) {

i++;

}

while (arr[j] > pivot) {

j--;

}

if (i <= j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

}

if (left < j) {

quicksort(arr, left, j);

}

if (i < right) {

quicksort(arr, i, right);

}

}

int main() {

int arr[] = { 5, 2, 9, 1, 5, 6 };

int n = sizeof(arr) / sizeof(arr[0]);

quicksort(arr, 0, n - 1);

printf("Sorted array: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**Date : 3/05/2023**

**1. Write a prog in c for check in array which is longgest pallindrom or not and print in output that array element ?**

#include <stdio.h>

#include <stdlib.h>

// Function to check whether a number is a palindrome or not

int is\_palindrome(int n) {

int rev = 0, temp = n;

while (temp > 0) {

rev = rev \* 10 + temp % 10;

temp /= 10;

}

return (n == rev);

}

//Function to find the longest palindrome(s) in an array of numbers

void longest\_palindrome(int arr[], int n) {

int longest\_len = 0;

int longest\_pals[n];

int num\_pals = 0;

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

if (is\_palindrome(arr[i])) {

int len = 0, temp = arr[i];

while (temp > 0) {

len++;

temp /= 10;

}

if (len > longest\_len) {

longest\_len = len;

num\_pals = 0;

longest\_pals[num\_pals++] = arr[i];

} else if (len == longest\_len) {

longest\_pals[num\_pals++] = arr[i];

}

}

}

if (num\_pals > 0) {

printf("Longest palindrome(s) in array:\n");

for (int i = 0; i < num\_pals; i++) {

printf("%d\n", longest\_pals[i]);

}

} else {

printf("No palindrome found in array.\n");

}

}

int main() {

int arr[] = {123, 121, 345, 23432, 678, 56765};

int n = sizeof(arr)/sizeof(arr[0]);

longest\_palindrome(arr, n);

return 0;

}

**Write a prog in c for Sorting of array ?**

#include <stdio.h>

void quicksort(int arr[], int left, int right) {

int i = left, j = right, pivot = arr[(left + right) / 2];

while (i <= j) {

while (arr[i] < pivot) {

i++;

}

while (arr[j] > pivot) {

j--;

}

if (i <= j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

}

if (left < j) {

quicksort(arr, left, j);

}

if (i < right) {

quicksort(arr, i, right);

}

}

int main() {

int arr[] = { 5, 2, 9, 1, 5, 6 };

int n = sizeof(arr) / sizeof(arr[0]);

quicksort(arr, 0, n - 1);

printf("Sorted array: ");

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

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**Date : 4/05/2023**

**1. Sort Array elements with less time complexity ?**

#include <stdio.h>

// Function to swap two elements

void swap(int\* a, int\* b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

// Function to partition the array and return the index of pivot element

int partition (int arr[], int low, int high)

{

int pivot = arr[high]; // pivot element

int i = (low - 1); // index of smaller element

for (int j = low; j <= high - 1; j++)

{

// If current element is smaller than or equal to pivot

if (arr[j] <= pivot)

{

i++; // increment index of smaller element

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

// Function to implement quicksort

void quicksort(int arr[], int low, int high)

{

if (low < high)

{

// pi is the partitioning index

int pi = partition(arr, low, high);

// Recursively sort elements before and after partition

quicksort(arr, low, pi - 1);

quicksort(arr, pi + 1, high);

}

}

int main()

{

int arr[] = {10, 7, 8, 9, 1, 5};

int n = sizeof(arr) / sizeof(arr[0]);

quicksort(arr, 0, n - 1);

printf("Sorted array: ");

for (int i = 0; i < n; i++)

printf("%d ", arr[i]);

return 0;

}

**2. Trapping rain water prog in c ?**

#include <stdio.h>

int findWater(int arr[], int n)

{

int left[n], right[n], i, maxWater = 0;

left[0] = arr[0];

for (i = 1; i < n; i++)

left[i] = (left[i-1] > arr[i]) ? left[i-1] : arr[i];

right[n-1] = arr[n-1];

for (i = n-2; i >= 0; i--)

right[i] = (right[i+1] > arr[i]) ? right[i+1] : arr[i];

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

maxWater += (left[i] < right[i]) ? (left[i] - arr[i]) : (right[i] - arr[i]);

return maxWater;

}

int main()

{

int arr[] = {3, 5, 1, 2, 1, 4};

int n = sizeof(arr)/sizeof(arr[0]);

printf("Maximum amount of water that can be trapped is %d\n", findWater(arr, n));

return 0;

}

**3. Find the equilibrium point of an array prog in c ?**

#include <stdio.h>

int findEquilibrium(int arr[], int n) {

int i, j, leftSum, rightSum;

for (i = 0; i < n; i++) {

leftSum = 0;

rightSum = 0;

for (j = 0; j < i; j++) {

leftSum += arr[j];

}

for (j = i + 1; j < n; j++) {

rightSum += arr[j];

}

if (leftSum == rightSum) {

return i;

}

}

return -1;

}

int main() {

int arr[] = {1, 4, 6, 2, 4, 0};

int n = sizeof(arr) / sizeof(arr[0]);

int equilibriumIndex = findEquilibrium(arr, n);

if (equilibriumIndex == -1) {

printf("There is no equilibrium point in the array\n");

} else {

printf("The equilibrium point is at index %d\n", equilibriumIndex);

}

return 0;

}

**4. Find leaders in array ?**

#include <stdio.h>

void printLeaders(int arr[], int size)

{

int i, j, max\_from\_right;

max\_from\_right = arr[size-1];

printf("%d ", max\_from\_right);

for(i = size-2; i >= 0; i--)

{

if(max\_from\_right < arr[i])

{

max\_from\_right = arr[i];

printf("%d ", max\_from\_right);

}

}

}

int main()

{

int arr[] = {16, 17, 4, 3, 5, 2};

int n = sizeof(arr)/sizeof(arr[0]);

printLeaders(arr, n);

return 0;

}

**Prime and Armstrong number prog in c ?**

#include <stdio.h>

#include <math.h>

// Function to check if a number is prime

int isPrime(int n) {

if (n < 2) {

return 0;

}

for (int i = 2; i <= sqrt(n); i++) {

if (n % i == 0) {

return 0;

}

}

return 1;

}

// Function to check if a number is an Armstrong number

int isArmstrong(int n) {

int numDigits = log10(n) + 1;

int sum = 0;

int temp = n;

while (temp > 0) {

int digit = temp % 10;

sum += pow(digit, numDigits);

temp /= 10;

}

return (sum == n);

}

// Function to concatenate two numbers and check if the result is prime and/or Armstrong

void concatenateAndCheck(int a, int b) {

int temp = b;

while (temp > 0) {

a \*= 10;

temp /= 10;

}

int result = a + b;

if (isPrime(result)) {

printf("%d is a prime number\n", result);

}

if (isArmstrong(result)) {

printf("%d is an Armstrong number\n", result);

}

}

int main() {

int size, i, j;

printf("Enter size of an array: ");

scanf("%d", &size);

int array[size];

printf("Enter array elements: ");

for (i = 0; i < size; i++) {

scanf("%d", &array[i]);

}

for (i = 0; i < size; i++) {

if (isPrime(array[i])) {

printf("%d is a prime number\n", array[i]);

}

if (isArmstrong(array[i])) {

printf("%d is an Armstrong number\n", array[i]);

}

for (j = i + 1; j < size; j++) {

concatenateAndCheck(array[i], array[j]);

}

}

return 0;

}

**Jumping numbers prog in c ?**