7. C program to find maximum in arr[] of size n

#include <stdio.h>

int largest(int arr[], int n)

{ int i;

int max = arr[0];

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

if (arr[i] > max)

max = arr[i];

return max;

}

int main()

{

int arr[] = { 10, 324, 45, 90, 9808 };

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

printf("Largest in given array is %d", largest(arr, n));

return 0;

}

8. c program to find maximum and minimum in an array

#include<stdio.h>

struct pair

{

int min;

int max;

};

struct pair getMinMax(int arr[], int n)

{

struct pair minmax;

int i;

/\*If there is only one element then return it as min and max both\*/

if (n == 1)

{

minmax.max = arr[0];

minmax.min = arr[0];

return minmax;

}

if (arr[0] > arr[1])

{

minmax.max = arr[0];

minmax.min = arr[1];

}

else

{

minmax.max = arr[1];

minmax.min = arr[0];

}

for (i = 2; i<n; i++)

{

if (arr[i] > minmax.max)

minmax.max = arr[i];

else if (arr[i] < minmax.min)

minmax.min = arr[i];

}

return minmax;

}

/\* Driver program to test above function \*/

int main()

{

int arr[] = {1000, 11, 445, 1, 330, 3000};

int arr\_size = 6;

struct pair minmax = getMinMax (arr, arr\_size);

printf("nMinimum element is %d", minmax.min);

printf("nMaximum element is %d", minmax.max);

getchar();

}

9. C program to implement binary search using recursion

#include <stdio.h>

int binarySearch(int arr[], int l, int r, int x)

{

// checking if there are elements in the subarray

if (r >= l) {

// calculating mid point

int mid = l + (r - l) / 2;

// If the element is present at the middle itself

if (arr[mid] == x)

return mid;

// If element is smaller than mid, then it can only

// be present in left subarray

if (arr[mid] > x) {

return binarySearch(arr, l, mid - 1, x);

}

// Else the element can only be present in right

// subarray

return binarySearch(arr, mid + 1, r, x);

}

return -1;

}

int main(void)

{

// taking a sorted array

int arr[] = { 2, 3, 4, 10, 40 };

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

// element to be searched

int x = 10;

// calling binary search

int index = binarySearch(arr, 0, size - 1, x);

if (index == -1) {

printf("Element is not present in array");

}

else {

printf("Element is present at index %d", index);

}

return 0;

}

11. C Program To Merge Two Array

#include <stdio.h>

int main()

{

int arr1size = 5, arr2size = 5, arr\_resultsize, i, j;

// elements of first Array

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

// elements of Second Array

int b[5] = { 6, 7, 8, 9, 10 };

// resultant Array Size Declaration

arr\_resultsize = arr1size + arr2size;

int c[arr\_resultsize];

// copying array 1 elements into an array

for (i = 0; i < arr1size; i++) {

c[i] = a[i];

}

// copying array 2 elements into an array

for (i = 0, j = arr1size;

j < arr\_resultsize && i < arr2size; i++, j++) {

c[j] = b[i];

}

for (i = 0; i < arr\_resultsize; i++) {

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

}

return 0;

}

12. C program to add two matrices

#include <stdio.h>

void add(int A[][N], int B[][N], int C[][N])

{

int i, j;

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

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

C[i][j] = A[i][j] + B[i][j];

}

void printmatrix(int D[][N])

{

int i, j;

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

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

printf("%d ", D[i][j]);

printf("\n");

}

}

int main()

{

int A[N][N] = { { 1, 1, 1, 1 },

{ 2, 2, 2, 2 },

{ 3, 3, 3, 3 },

{ 4, 4, 4, 4 } };

int B[N][N] = { { 1, 1, 1, 1 },

{ 2, 2, 2, 2 },

{ 3, 3, 3, 3 },

{ 4, 4, 4, 4 } };

int C[N][N];

int i, j;

printf("Matrix A is \n");

printmatrix(A);

printf("Matrix B is \n");

printmatrix(B);

add(A, B, C);

printf("Result matrix is \n");

printmatrix(C);

13. C implementation to check if a given string is palindrome or not

#include <stdio.h>

#include <string.h>

int main()

{

char str[] = { "abbba" };

// Start from first and

// last character of str

int l = 0;

int h = strlen(str) - 1;

// Keep comparing characters

// while they are same

while (h > l) {

if (str[l++] != str[h--]) {

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

return 0;

// will return from here

}

}

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

return 0;

}

14.

15. C program to print ASCII Value of Character

#include <stdio.h>

int main()

{

char c = 'k';

// %d displays the integer value of

// a character

// %c displays the actual character

printf("The ASCII value of %c is %d", c, c);

return 0;

}

16. C program to demonstrate the area and perimeter of rectangle

#include <stdio.h>

int main()

{

int l = 10, b = 10;

printf("Area of rectangle is : %d", l \* b);

printf("\nPerimeter of rectangle is : %d", 2 \* (l + b));

return 0;

17. C Program to Store Information of Students Using Structure

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Student {

char\* name;

int roll\_number;

int age;

double total\_marks;

};

int main()

{

int i = 0, n = 5;

struct Student student[n];

student[0].roll\_number = 1;

student[0].name = "Geeks1";

student[0].age = 12;

student[0].total\_marks = 78.50;

student[1].roll\_number = 5;

student[1].name = "Geeks5";

student[1].age = 10;

student[1].total\_marks = 56.84;

student[2].roll\_number = 2;

student[2].name = "Geeks2";

student[2].age = 11;

student[2].total\_marks = 87.94;

student[3].roll\_number = 4;

student[3].name = "Geeks4";

student[3].age = 12;

student[3].total\_marks = 89.78;

student[4].roll\_number = 3;

student[4].name = "Geeks3";

student[4].age = 13;

student[4].total\_marks = 78.55;

// Print the Students information

printf("Student Records:\n\n");

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

{

printf("\tName = %s\n", student[i].name);

printf("\tRoll Number = %d\n", student[i].roll\_number);

printf("\tAge = %d\n", student[i].age);

printf("\tTotal Marks = %0.2f\n\n", student[i].total\_marks);

}

return 0;

}

18. C Program to print the Fibonacci series using recursion

#include <stdio.h>

int prev1 = 1;

int prev2 = 0;

// recursive function to print the fibonacci series

void fib(int n)

{

if (n < 3) {

return;

}

int fn = prev1 + prev2;

prev2 = prev1;

prev1 = fn;

printf("%d ", fn);

return fib(n - 1);

}

void printFib(int n)

{

// when the number of terms is less than 1

if (n < 1) {

printf("Invalid number of terms\n");

}

// when the number of terms is 1

else if (n == 1) {

printf("%d ", 0);

}

// when the number of terms is 2

else if (n == 2) {

printf("%d %d", 0, 1);

}

// number of terms greater than 2

else {

printf("%d %d ", 0, 1);

fib(n);

}

return;

}

int main()

{

int n = 9;

printFib(n);

return 0;

}

19. C program to find LCM of two numbers

#include <stdio.h>

int main()

{

int x = 15, y = 25, max;

max = (x > y) ? x : y;

// While loop to check if max variable

// is divisible by x and y

while (1) {

if (max % x == 0 && max % y == 0) {

printf("The LCM of %d and %d is %d.", x, y,

max);

break;

}

++max;

}

return 0;

}

20. C Program to make a Simple Calculator Using switch case

#include <stdio.h>

#include <stdlib.h>

int main()

{

char ch;

double a, b;

while (1) {

printf("Enter an operator (+, -, \*, /), "

"if want to exit press x: ");

scanf(" %c", &ch);

// to exit

if (ch == 'x')

exit(0);

printf("Enter two first and second operand: ");

scanf("%lf %lf", &a, &b);

switch (ch) {

// For Addition

case '+':

printf("%.1lf + %.1lf = %.1lf\n", a, b, a + b);

break;

// For Subtraction

case '-':

printf("%.1lf - %.1lf = %.1lf\n", a, b, a - b);

break;

// For Multiplication

case '\*':

printf("%.1lf \* %.1lf = %.1lf\n", a, b, a \* b);

break;

// For Division

case '/':

printf("%.1lf / %.1lf = %.1lf\n", a, b, a / b);

break;

// If operator doesn't match any case constant

default:

printf(

"Error! please write a valid operator\n");

}

printf("\n");

}

}

21. C program to reverse a linked list

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

/\* Function to reverse the linked list \*/

static void reverse(struct Node\*\* head\_ref)

{

struct Node\* prev = NULL;

struct Node\* current = \*head\_ref;

struct Node\* next = NULL;

while (current != NULL) {

// Store next

next = current->next;

// Reverse current node's pointer

current->next = prev;

// Move pointers one position ahead.

prev = current;

current = next;

}

\*head\_ref = prev;

}

/\* Function to push a node \*/

void push(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node

= (struct Node\*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node\* head)

{

struct Node\* temp = head;

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

}

int main()

{

struct Node\* head = NULL;

push(&head, 20);

push(&head, 4);

push(&head, 15);

push(&head, 85);

printf("Given linked list\n");

printList(head);

reverse(&head);

printf("\nReversed linked list \n");

printList(head);

getchar();

}

23. // C program to reverse a string using stack

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Stack {

int top;

unsigned capacity;

char\* array;

};

struct Stack\* createStack(unsigned capacity)

{

struct Stack\* stack

= (struct Stack\*)malloc(sizeof(struct Stack));

stack->capacity = capacity;

stack->top = -1;

stack->array

= (char\*)malloc(stack->capacity \* sizeof(char));

return stack;

}

// Stack is full when top is equal to the last index

int isFull(struct Stack\* stack)

{

return stack->top == stack->capacity - 1;

}

// Stack is empty when top is equal to -1

int isEmpty(struct Stack\* stack)

{

return stack->top == -1;

}

void push(struct Stack\* stack, char item)

{

if (isFull(stack))

return;

stack->array[++stack->top] = item;

}

char pop(struct Stack\* stack)

{

if (isEmpty(stack))

return INT\_MIN;

return stack->array[stack->top--];

}

void reverse(char str[])

{

// Create a stack of capacity

// equal to length of string

int n = strlen(str);

struct Stack\* stack = createStack(n);

// Push all characters of string to stack

int i;

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

push(stack, str[i]);

// Pop all characters of string and

// put them back to str

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

str[i] = pop(stack);

}

int main()

{

char str[] = "GeeksQuiz";

reverse(str);

printf("Reversed string is %s", str);

return 0;

24. #include <stdio.h>

// Function to sort the numbers using pointers

void sort(int n, int\* ptr)

{

int i, j, t;

// Sort the numbers using pointers

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

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

if (\*(ptr + j) < \*(ptr + i)) {

t = \*(ptr + i);

\*(ptr + i) = \*(ptr + j);

\*(ptr + j) = t;

}

}

}

// print the numbers

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

printf("%d ", \*(ptr + i));

}

int main()

{

int n = 5;

int arr[] = { 0, 23, 14, 12, 9 };

sort(n, arr);

return 0;

}

25. C program to sort array elements in descending order

#include <stdio.h>

int main()

{

int a[5] = { 45, 22, 100, 66, 37 };

int n = 5, i, j, t = 0;

// iterates the array elements

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

// iterates the array elements from index 1

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

// comparing the array elements, to set array

// elements in descending order

if (a[i] < a[j]) {

t = a[i];

a[i] = a[j];

a[j] = t;

}

}

}

// printing the output

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

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

}

return 0;

}

26. C program to find the length of string

#include <stdio.h>

#include <string.h>

int main()

{

char Str[1000];

int i;

printf("Enter the String: ");

scanf("%s", Str);

for (i = 0; Str[i] != '\0'; ++i);

printf("Length of Str is %d", i);

return 0;

}

27. C program to check whether a number can be expressed as sum of two prime numbers

#include <stdio.h>

// Function to check prime number

int isPrime(int n)

{

int i, isPrime = 1;

// 0 and 1 are not prime numbers

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

isPrime = 0;

}

else {

for (i = 2; i <= n / 2; ++i) {

if (n % i == 0) {

isPrime = 0;

break;

}

}

}

return isPrime;

}

int main()

{

int n = 7, i, flag = 0;

for (i = 2; i <= n / 2; ++i) {

// condition for i to be a

// prime number

if (isPrime(i) == 1) {

// condition for n-i to

// be a prime number

if (isPrime(n - i) == 1) {

printf("Yes\n");

return 0;}}

}

printf("No\n");

return 0;

}

return 0;

}

28. C program to find the length of string

#include <stdio.h>

#include <string.h>

int main()

{

char Str[1000];

int i;

printf("Enter the String: ");

scanf("%s", Str);

for (i = 0; Str[i] != '\0'; ++i);

printf("Length of Str is %d", i);

return 0;

}

29. C program to multiply two matrices

#include <stdio.h>

#include <stdlib.h>

#define R1 2 // number of rows in Matrix-1

#define C1 2 // number of columns in Matrix-1

#define R2 2 // number of rows in Matrix-2

#define C2 3 // number of columns in Matrix-2

void multiplyMatrix(int m1[][C1], int m2[][C2])

{

int result[R1][C2];

printf("Resultant Matrix is:\n");

for (int i = 0; i < R1; i++) {

for (int j = 0; j < C2; j++) {

result[i][j] = 0;

for (int k = 0; k < R2; k++) {

result[i][j] += m1[i][k] \* m2[k][j];

}

printf("%d\t", result[i][j]);

}

printf("\n");

}

}

int main()

{

int m1[R1][C1] = { { 1, 1 }, { 2, 2 } };

int m2[R2][C2] = { { 1, 1, 1 }, { 2, 2, 2 } };

if (C1 != R2) {

printf("The number of columns in Matrix-1 must be equal to the number of rows in Matrix-2\n");

printf("Please update value according to your array dimension in #define section\n");

exit(EXIT\_FAILURE);

}

multiplyMatrix(m1, m2);

return 0;

}

30. #include <stdio.h>

// Function to print the alphabet in lower case

void lowercaseAlphabets()

{

// lowercase

for (int c = 97; c <= 122; ++c)

cout << char(c) << " "; // Convert ASCII value to character

cout << endl;

}

// Function to print the alphabet in upper case

void uppercaseAlphabets()

{

// uppercase

for (int c = 65; c <= 90; ++c)

cout << char(c) << " "; // Convert ASCII value to character

cout << endl;

}

int main()

{

cout << "Uppercase Alphabets" << endl;

uppercaseAlphabets();

cout << "Lowercase Alphabets " << endl;

lowercaseAlphabets();

return 0;

}