**Assigment:1 DATE: 02/01/2024**

**AIM:Add comments to program.**

**Pratical :1**

**#include <stdio.h>**

*// Swap Function Definition*

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

**int temp = \*a;**

**\*a = \*b;**

**\*b = temp;**

**}**

*// Main Function*

**int main() {**

**int x = 5, y = 10;**

*// Print values before swap*

**printf("Before Swap: x = %d, y = %d\n", x, y);**

*// Call Swap Function*

**swap(&x, &y);**

*// Print values after swap*

**printf("After Swap: x = %d, y = %d\n", x, y);**

**return 0;**

**}**

**Pratical :2**

**#include <stdio.h>**

*// main function*

**int main() {**

*// Initialize an integer array*

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

*// Declare and initialize an integer pointer to the first element of the array*

**int \*ptr = arr;**

*// Initialize a variable to store the sum of array elements*

**int sum = 0;**

*// Iterate through the array using the pointer and calculate the sum*

**for (int i = 0; i < 5; ++i) {**

**sum += \*ptr;** *// Accumulate the value pointed to by ptr*

**ptr++;** *// Move the pointer to the next element*

}

*// Print the sum of array elements*

**printf("Sum of elements: %d\n", sum);**

**return 0;**

**}**

**Pratical :3**

*// Header file*

**#include <stdio.h>**

*// Main function*

**int main() {**

**int x = 10;**

**printf("Address of x: %p\n", (void\*)&x);** *// Prints the memory address of variable x*

**return 0;**

**}**

**Pratical :4**

**#include <stdio.h>**

*// Function to calculate the length of a string*

**int stringLength(char \*str) {**

**int length = 0;** *// Initialize length to zero*

**while (\*str != '\0') {**

**length++;** *// Increment length for each character*

**str++;**  *// Move to the next character in the string*

**}**

**return length;** *// Return the final length*

**}**

*// Main function*

**int main() {**

**char str[] = "Hello, World!";** *// Declare and initialize a string*

**printf("Length of the string: %d\n", stringLength(str));** *// Print the length of the string*

**return 0;** *// Indicate successful program execution*

**}**

**Pratical :5**

**#include <stdio.h>**

**#include <stdlib.h>**

*// Dynamic memory allocation example*

**int main() {**

*// Declare a pointer to int*

**int \*ptr;**

*// Allocate memory for an integer*

**ptr = (int \*)malloc(sizeof(int));** *// Allocating memory of size int*

*// Check if memory allocation is successful*

**if (ptr == NULL) {**

**printf("Memory allocation failed.\n");** *// Error message if allocation fails*

**return 1;**

**}**

*// Store a value in the dynamically allocated memory*

**\*ptr = 10;**

*// Print the value stored in the allocated memory*

**printf("Value stored at dynamically allocated memory: %d\n", \*ptr);**

*// Free the allocated memory to avoid memory leaks*

**free(ptr);**

**return 0;**

**}**

**Pratical :6**

**#include <stdio.h>**

*// Function to find the largest element in an array*

**int findLargest(int \*arr, int size) {**

**int max = \*arr;** *// Initialize max with the first element of the array*

**for (int i = 1; i < size; ++i) {**

**if (\*(arr + i) > max) {**

**max = \*(arr + i);** *// Update max if a larger element is found*

**}**

**}**

**return max;**

**}**

*// Main function*

**int main() {**

**int arr[] = {5, 12, 3, 8, 15};**

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

*// Print the largest element in the array*

**printf("Largest element in the array: %d\n", findLargest(arr, size));**

**return 0;**

**}**

**Pratical :7**

**#include <stdio.h>**

*// Function to display elements of an array*

**void displayArray(int \*arr, int size) {**

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

**printf("%d\n", \*(arr + i));** *// Display each element in a new line*

**}**

**printf("\n");**

**}**

*// main function*

**int main() {**

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

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

**printf("Array elements:\n");**

**displayArray(arr, size);**

**return 0;**

**}**

**Pratical :8**

**#include <stdio.h>**

*// Function to check whether a number is even or odd*

**int checkEvenOdd(int \*num) {**

**if (\*num % 2 == 0) {**

**return 1; // Even**

**} else {**

**return 0; // Odd**

**}**

**}**

*// Main function*

**int main() {**

**int x = 7;** *// Input number*

**if (checkEvenOdd(&x)) {**

**printf("%d is even.\n", x);** *// Print if the number is even*

**} else {**

**printf("%d is odd.\n", x);** *// Print if the number is odd*

**}**

**return 0;**

**}**

**Pratical :9**

**#include <stdio.h>**

*// Function to square the value pointed to by num*

**void squareValue(int \*num) {**

**\*num = (\*num) \* (\*num);** *// Squaring the value using pointer dereferencing*

**}**

**int main() {**

**int x = 5;** *// Initializing variable x with the value 5*

**printf("Before: %d\n", x);** *// Printing the value of x before the function call*

**squareValue(&x);** *// Calling the squareValue function, passing the address of x*

**printf("After: %d\n", x);** *// Printing the value of x after the function call*

**return 0;**

**}**

**Pratical :10**

**#include <stdio.h>**

*// Function to add two matrices*

**void addMatrices(int \*mat1, int \*mat2, int \*result, int rows, int cols) {**

**for (int i = 0; i < rows; ++i) {**

**for (int j = 0; j < cols; ++j) {**

**\*(result + i \* cols + j) = \*(mat1 + i \* cols + j) + \*(mat2 + i \* cols + j);**

**}**

**}**

**}**

*// Function to display a matrix*

**void displayMatrix(int \*mat, int rows, int cols) {**

**for (int i = 0; i < rows; ++i) {**

**for (int j = 0; j < cols; ++j) {**

**printf("%d ", \*(mat + i \* cols + j));**

**}**

**printf("\n");**

**}**

**}**

**int main() {**

*// Matrix initialization and addition*

**int mat1[2][2] = {{1, 2}, {3, 4}};  
  
        int mat2[2][2] = {{5, 6}, {7, 8}};  
  
        int result[2][2];  
 int rows = 2, cols = 2;**

**addMatrices(&mat1[0][0],  
&mat2[0][0], &result[0][0], rows, cols);**

*// Displaying matrices and result*

**printf("Matrix 1:\n");  
  
        displayMatrix(&mat1[0][0],  
rows, cols);  
  
 printf("\nMatrix 2:\n");  
  
        displayMatrix(&mat2[0][0],  
rows, cols);  
 printf("\nSum of  
Matrices:\n");  
  displayMatrix(&result[0][0],  
rows, cols);**

**return 0;**

**}**