**DS Lab Manual**

Problem 1 :- To implement basic operations on arrays and understand their memory representation.

Code :-

#include <stdio.h>

void display(int arr[], int size) {

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

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

}

printf("\n");

}

int main() {

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

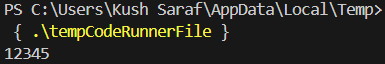
int size = sizeof(arr) / sizeof(arr[0]);

display(arr, size);

return 0;

}

Output :-



Problem 2 :- To implement basic operations on arrays two sum in array.

Code :-

#include <stdio.h>

void twoSum(int arr[], int size, int target) {

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

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

if(arr[i] + arr[j] == target) {

printf("Pair found: %d + %d = %d\n", arr[i], arr[j], target);

return;

}

}

}

printf("No pair found\n");

}

int main() {

int arr[] = {2, 7, 11, 15};

int target = 9;

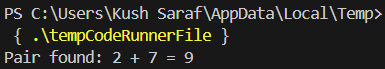
int size = sizeof(arr) / sizeof(arr[0]);

twoSum(arr, size, target);

return 0;

}

Output :-



Problem 3 :- To implement basic operations on arrays using recursion.

Code :-

#include <stdio.h>

int sumArray(int arr[], int size) {

if (size == 0)

return 0;

return arr[size - 1] + sumArray(arr, size - 1);

}

int main() {

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

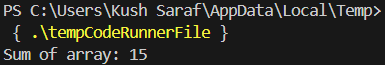
int size = sizeof(arr) / sizeof(arr[0]);

printf("Sum of array: %d\n", sumArray(arr, size));

return 0;

}

Output :-



Problem 4 :- To understand and implement recursive functions for problem-solving.

Code :-

#include <stdio.h>

int factorial(int n) {

if(n <= 1)

return 1;

return n \* factorial(n - 1);

}

int main() {

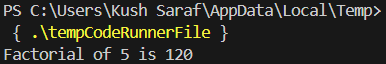
int number = 5;

printf("Factorial of %d is %d\n", number, factorial(number));

return 0;

}

Output :-



Problem 5 :- To implement stack data structures and apply them in expression conversion and evaluation.

Code :-

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct Stack{

int top;

int items[MAX];

}Stack;

void push(Stack \*stack, int value) {

if(stack->top < MAX - 1) {

stack->items[++stack->top] = value;

}

}

int pop(Stack \*stack) {

if(stack->top >= 0) {

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

}

}

int main() {

Stack stack;

stack.top = -1;

push(&stack, 10);

push(&stack, 20);

printf("Popped: %d\n", pop(&stack));

return 0;

}

Output :-

