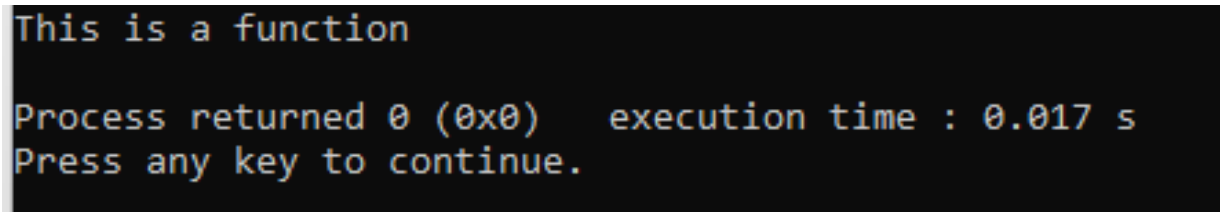
Name: Nusrat Jahan Chaiti
ID: 2023-1-60-255
Course Code : CSE 103
Course Title: Structured Programming Language
Section: 11

Q1. Function to print a custom message.

Answer:

```
#include <stdio.h>

void printMessage() {
printf("This is a function\n");
}
int main() {
printMessage();
return 0;
}
```

Output:



Q2. Function to print an input character value.

Answer:

```
#include <stdio.h>

void printValue(int num, char ch) {
printf("Value received from main: %d\n", num);
printf("Value received from main: %c\n", ch);
}

int main() {
int num = 3;
char ch = 'A';
printValue(num, ch);

return 0;
}
```

Output:

```
Value received from main: 3
Value received from main: A
```

Q3. Function to calculate the sum of n numbers coming from the console.

Answer:

```
#include <stdio.h>

int calculateSum(int n) {
int num, sum = 0;

for (int i = 0; i < n; i++) {
scanf("%d", &num);
sum += num;
}
printf("Sum In Function: %d\n", sum);

return sum;
}

int main() {
int n;
printf("Enter the number of integers to add: ");
scanf("%d", &n);

int result = calculateSum(n);
printf("Sum in Main: %d\n", result);

return 0;
}
```

Output:

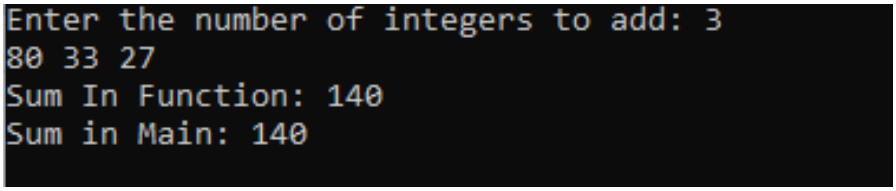
```
Enter the number of integers to add:  3
80 33 27
Sum In Function: 140
Sum in Main: 140
```

Q4. Function to calculate the sum of n numbers coming from the console and stored in an array.

Answer:

```
#include <stdio.h>

int calculateSum(int arr[], int n) {
int sum = 0;
for (int i = 0; i < n; i++) {
sum += arr[i];
}
return sum;
}

int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);

int arr[n];

for (int i = 0; i <n; i++) {
scanf("%d", &arr[i]);
}
int result = calculateSum(arr, n);
printf("Sum In Function: %d\n", result);
printf("Sum in Main: %d\n", result);

return 0;
}
```

Output:



Q5. Function to swap two numbers. (Restriction: Pass by value)

Answer:

```
#include <stdio.h>

void swap(int x, int y) {
int temp = x;
x = y;
y = temp;
}

int main() {
int a = 10, b = 20;

printf("Before swap: a = %d, b = %d\n", a, b);
swap(a, b);
printf("After swap: a = %d, b = %d\n", a, b);

return 0;
}
```

Output:

```
Before swap: a = 10, b = 20
After swap: a = 10, b = 20
```

Q6. Function to swap two numbers. (Restriction: Pass by reference)

Answer:

```
#include <stdio.h>

void swap(int *x, int *y) {
int temp = *x;
*x = *y;
*y = temp;
}

int main() {
int a = 10, b = 20;

printf("Before swap: a = %d, b = %d\n", a, b);
swap(&a, &b);
printf("After swap: a = %d, b = %d\n", a, b);

return 0;
}
```

Output:

```
Before swap: a = 10, b = 20
After swap: a = 20, b = 10
```

Q7. Function to determine only even numbers in an array of input integers.

Answer:

```
#include <stdio.h>

void findEvenNumbers(int arr[], int n) {
printf("Even numbers in the array: ");
for (int i = 0; i < n; i++) {
if (arr[i] % 2 == 0) {
printf("%d ", arr[i]);
}
}
printf("\n");
}

int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);

int arr[n];
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
findEvenNumbers(arr, n);

return 0;
}
```

Output:

```
Enter the size of the array: 4
45 33 0 256
Even numbers in the array: 0 256
```

Q8. Function that finds and returns the minimum value in an array

Answer:

```
#include <stdio.h>

int findMinValue(int arr[], int n) {
int minValue = arr[0];

for (int i = 1; i < n; i++) {
if (arr[i] < minValue) {
minValue = arr[i];
}
}

return minValue;
}

int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);

int arr[n];

for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}

int minValue = findMinValue(arr, n);
printf("Minimum value in the array: %d\n", minValue);

return 0;
}
```

Output:

```
Enter the size of the array: 5
157 -28 -37 26 10
Minimum value in the array: -37
```

Q9. Function that multiplies the array elements by 2 and returns the array

Answer:

```
#include <stdio.h>

void multiplyByTwo(int arr[], int n) {
for (int i = 0; i < n; i++) {
arr[i] *= 2;
}
}

int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);

int arr[n];
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
multiplyByTwo(arr, n);

for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}

return 0;
}
```

Output:

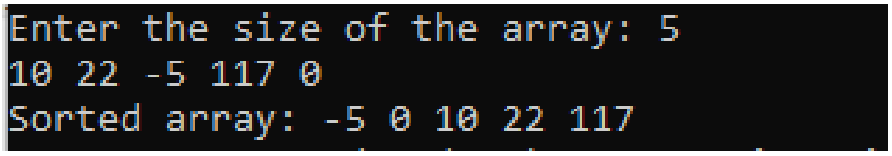
```
Enter the size of the array: 5
157 -28 -37 26 10
314 -56 -74 52 20
```

Q10. Function to sort and return an input array in ascending order

Answer:

```
#include <stdio.h>

void sortArray(int arr[], int n) {
int temp;
for (int i = 0; i < n; i++) {
for (int j = i + 1; j < n; j++) {
if (arr[i] > arr[j]) {
// Swap the elements at indices i and j
temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
}
}
}

int main() {
int n;
printf("Enter the size of the array: ");
scanf("%d", &n);

int arr[n];
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}

sortArray(arr, n);

printf("Sorted array: ");
for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}

return 0;
}
```

Output:



Q11. Function "IsPrime()" to determine whether a number is prime or not.

Answer:

```
#include <stdio.h>

int IsPrime(int num) {
if (num < 2) {
return 0;
}
for (int i = 2; i <= num / 2; i++) {
if (num % i == 0) {
return 0;
}
}
return 1;
}

int main() {
int n;
printf("Enter a positive integer: ");
scanf("%d", &n);

if (IsPrime(n)) {
printf("%d is a prime number\n", n);
} else {
printf("%d is not a prime number\n", n);
}

return 0;
}
```

Output:

```
Enter a positive integer: 2
2 is a prime number
```

Q12. Function "GeneratePrime()" to compute the prime numbers less than N, where N is an input integer. GeneratePrime() uses IsPrime() to check whether a number is prime or not.

Answer:

```
#include <stdio.h>

int IsPrime(int num) {
if (num < 2) {
return 0;
}
for (int i = 2; i <= num / 2; i++) {
if (num % i == 0) {
return 0;
}
}
return 1;
}

void GeneratePrime(int N) {
printf("Prime numbers less than %d are: ", N);

for (int i = 2; i < N; i++) {
if (IsPrime(i)) {
printf("%d ", i);
}
}
printf("\n");
}

int main() {
int n;
printf("Enter a positive integer: ");
scanf("%d", &n);

GeneratePrime(n);

return 0;
}
```

Output:

```
Enter a positive integer: 10
Prime numbers less than 10 are: 2 3 5 7
```

Q13. Function "GenNthPrime()" to compute the Nth prime number, where N is an integer input.

Answer:

```
#include <stdio.h>

int isPrime(int n) {
if (n <= 1) {
return 0;
}
for (int i = 2; i * i <= n; i++) {
if (n % i == 0) {
return 0;
}
}
return 1;
}
int genNthPrime(int n) {
int count = 0, i = 2;
while (count < n) {
if (isPrime(i)) {
count++;
}
i++;
}
return i - 1;
}
int main() {
int n;
printf("Enter the value of N: ");
scanf("%d", &n);
int nthPrime = genNthPrime(n);
printf("The %dth prime number is %d\n", n, nthPrime);
return 0;
}
```

Output:

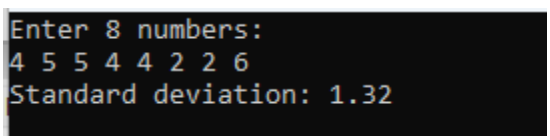
```
Enter the value of N: 40
The 40th prime number is 173
```

Q14. Implement the following functions and calculate standard deviation of an array whose values come from the terminal-
TakeInput()
CalcMean(array, num_of_elem) Calc_Std_deviation (array, num_of_elem)

Answer:

```
#include <stdio.h>
#include <math.h>

void TakeInput(int arr[], int n) {
  printf("Enter %d numbers:\n", n);
  for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
}
double CalcMean(int arr[], int n) {
  double sum = 0.0;
  for (int i = 0; i < n; i++) {
    sum += arr[i];
  }
  return sum / n;
}
double CalcStdDeviation(int arr[], int n) {
  double mean = CalcMean(arr, n);
  double sum = 0.0;
  for (int i = 0; i < n; i++) {
    sum += pow(arr[i] - mean, 2);
  }
  return sqrt(sum / n);
}

int main() {
  int n = 8;
  int arr[n];
  TakeInput(arr, n);
  double std_deviation = CalcStdDeviation(arr, n);
  printf("Standard deviation: %.2f\n", std_deviation);
  return 0;
}
```

Output:



Q15. Function find_substr() that takes two string arrays (a, b) as parameters, returns 1 if string b is found anywhere in string a, or returns-1 if no match is found.

Answer:

```
#include <stdio.h>
#include <string.h>

int find_substr(char a[], char b[]) {
int len_a = strlen(a);
int len_b = strlen(b);

for (int i = 0; i <= len_a - len_b; i++) {
int j;
for (j = 0; j < len_b; j++) {
if (a[i + j] != b[j])
break;
}
if (j == len_b)
return 1;
}
return -1;
}
int main() {
char a[100], b[100];
printf("Enter string a: ");
scanf("%s", a);
printf("Enter string b: ");
scanf("%s", b);

int result = find_substr(a, b);
if (result == 1) {
printf("%s is found in %s\n", b, a);
} else {
printf("%s is not found in %s\n", b, a);
}
return 0;
}
```

Output:

```
Enter string a: madam adam
Enter string b: adam is found in madam
```

Q16. Function find_ substr() that takes two string arrays (a, b) as parameters, uses function str_ length() to determine the lengths of the strings, and then looks for the smaller string anywhere in the bigger string. It returns 1 if the substring is found, or returns-1 if no match is found.

Answer:

```
#include <stdio.h>

int str_length(char str[]) {
int len = 0;
while (str[len] != '\0') {
len++;
}
return len;
}
int find_substr(char a[], char b[]) {
int len_a = str_length(a);
int len_b = str_length(b);

if (len_a < len_b) {
return -1;
}

for (int i = 0; i <= len_a - len_b; i++) {
int j;
for (j = 0; j < len_b; j++) {
if (a[i + j] != b[j]) {
break;
}
}
if (j == len_b) {
return 1;
}
}

return -1;
}
int main() {
char a[100], b[100];

printf("Enter string a: ");
scanf("%s", a);

printf("Enter string b: ");
scanf("%s", b);

int result = find_substr(a, b);
if (result == 1) {
printf("String b is found in string a.\n");
} else {
printf("String b is not found in string a.\n");
}

return 0;
}
```

Output ;

```
Enter string a: madam adam
Enter string b: adam is found in madam
```

Q17. Program that continuously takes two positive integers as inputs and uses two functions to find their GCD (greatest common divisor) and LCM (least common multiple). Both functions take parameters and returns desired values.

Answer:

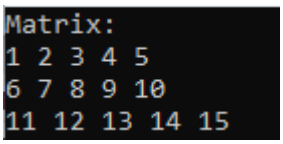
```
#include <stdio.h>
int findGCD(int num1, int num2) {
int gcd = 1, i;
for(i=1; i<=num1 && i<=num2; ++i) {
if(num1%i==0 && num2%i==0) {
gcd = i;
}
}
return gcd;
}
int findLCM(int num1, int num2) {
int lcm = (num1 > num2) ? num1 : num2;
while(1) {
if(lcm%num1==0 && lcm%num2==0) {
return lcm;
}
lcm++;
}
}

int main() {
int num1, num2;

while(1) {
printf("Enter two positive integers: ");
scanf("%d %d", &num1, &num2);

if(num1 <= 0 || num2 <= 0) {
printf("Invalid input! Both numbers must be positive integers.\n");
continue;
}
int gcd = findGCD(num1, num2);
int lcm = findLCM(num1, num2);
printf("GCD of %d and %d is %d\n", num1, num2, gcd);
printf("LCM of %d and %d is %d\n", num1, num2, lcm);
}
return 0;
}
```

```
Enter two positive integers: 5 7
GCD of 5 and 7 is 1
LCM of 5 and 7 is 35
```

Q18. Program that implements function to perform operations on a 3X5 matric

Answer:

```
#include <stdio.h>
#define ROWS 3
#define COLS 5

void printMatrix(int matrix[ROWS][COLS]) {
printf("Matrix:\n");
for (int i = 0; i < ROWS; i++) {
for (int j = 0; j < COLS; j++) {
printf("%d ", matrix[i][j]);
}
printf("\n");
}
}
void multiplyByTwo(int matrix[ROWS][COLS]) {
for (int i = 0; i < ROWS; i++) {
for (int j = 0; j < COLS; j++) {
matrix[i][j] *= 2;
}
}
}

void addRowToMatrix(int matrix[ROWS][COLS], int row[COLS], int rowIndex) {
for (int j = 0; j < COLS; j++) {
matrix[rowIndex][j] += row[j];
}
}

int main() {
int matrix[ROWS][COLS] = {
{1, 2, 3, 4, 5},
{6, 7, 8, 9, 10},
{11, 12, 13, 14, 15}
};
printMatrix(matrix);
multiplyByTwo(matrix);
printMatrix(matrix);
int row[COLS] = {1, 1, 1, 1, 1};
addRowToMatrix(matrix, row, 0);
printMatrix(matrix);

return 0;
}
```



Q19. Program that implements function to perform operations on a MXN matrix:

Answer:

```
#include <stdio.h>

#define M 3
#define N 3
#define P 3

void matrix_multiply(int A[][N], int B[][P], int C[][P])
{
int i, j, k;
for(i = 0; i < M; i++)
{
for(j = 0; j < P; j++)
{
C[i][j] = 0;
for(k = 0; k < N; k++)
{
C[i][j] += A[i][k] * B[k][j];
}
}
}
}
void print_matrix(int matrix[][P])
{
int i, j;
for(i = 0; i < M; i++)
{
for(j = 0; j < P; j++)
{
printf("%d ", matrix[i][j]);
}
printf("\n");
}
}
int main()
{
int A[M][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
int B[N][P] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
int C[M][P];
matrix_multiply(A, B, C);
printf("Matrix A:\n");
print_matrix(A);

printf("\nMatrix B:\n");
print_matrix(B);
printf("\nMatrix C = A * B:\n");
print_matrix(C);
return 0;
}
```

```
Matrix A:
1 2 3
4 5 6
7 8 9

Matrix B:
9 8 7
6 5 4
3 2 1

Matrix C = A * B:
30 24 18
84 69 54
138 114 90
```

Q20. Program to convert a positive integer to another base using the following functions-
1. Get Number_And_Base () : Takes number to be converted (N) and base value (B) from user. Base must be between 2 and 16.
II. Convert_Number (): Does the conversion
III. Show Converted_Number(): Displays the converted value.

Answer:

```
#include <stdio.h>
#include <stdlib.h>

void getNumberAndBase(int *n, int *b) {
printf("Enter a positive integer: ");
scanf("%d", n);
printf("Enter the base to convert to (between 2 and 16): ");
scanf("%d", b);
}
void convertNumber(int n, int b, char *converted) {
int quotient = n;
int remainder;
int i = 0;

while (quotient != 0) {
remainder = quotient % b;
if (remainder < 10) {
converted[i] = remainder + '0';
} else {
converted[i] = remainder + 55;
}
i++;
quotient = quotient / b;
}
int j;
char temp;
for (j = 0; j < i/2; j++) {
temp = converted[j];
converted[j] = converted[i-j-1];
converted[i-j-1] = temp;
}
}

void showConvertedNumber(char *converted) {
printf("The converted number is: %s\n", converted);
}

int main() {
int n, b;
char converted[100];
getNumberAndBase(&n, &b);
convertNumber(n, b, converted);
showConvertedNumber(converted);
return 0;
}
```

```
Enter a positive integer: 100 8
Enter the base to convert to (between 2 and 16): The converted number is: 144
```