Assignment

Week 1

Tarun Tom – B230589EC

2025

# Part 1

## Q1

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

int main() {

    // Define items and prices

    string items[5] = { "Tomatoes", "Potatoes", "Onions", "Carrots", "Spinach" };

    double prices[5] = { 200, 180, 260, 190, 110 }; // per kg

    double quantities[5];  //kg

    double total = 0.0;

    // Get quantities from the user

    cout << "Enter the quantity (in kg) for each item:\n";

    for (int i = 0; i < 5; i++) {

        cout << items[i] << " (Rs " << prices[i] << "/kg): ";

        cin >> quantities[i];

        total += prices[i] \* quantities[i];

    }

    // Print the bill

    cout << "\n========== BILL ==========\n";

    cout << setw(15) << left << "Item"

        << setw(10) << "Price"

        << setw(10) << "Quantity"

        << setw(10) << "Total" << endl;

    cout << setprecision(2) << fixed;

    for (int i = 0; i < 5; i++) {

        cout << setw(15) << left << items[i]

            << setw(10) << prices[i]

            << setw(10) << quantities[i]

            << setw(10) << prices[i] \* quantities[i] << endl;

    }

    cout << "==========================\n";

    cout << setw(35) << "Grand Total: Rs" << total << endl;

    return 0;

}

### Output

## 

## Q2

#include <iostream>

using namespace std;

int main() {

    int marks;

    char grade;

    cout << "Enter the marks obtained (0-100): ";

    cin >> marks;

    // Determine the grade based on marks using switch

    switch (marks / 10) {

    case 10:

    case 9:

        grade = 'S';

        break;

    case 8:

        grade = 'A';

        break;

    case 7:

        grade = 'B';

        break;

    case 6:

        grade = 'C';

        break;

    case 5:

        grade = 'D';

        break;

    case 4:

        grade = 'E';

        break;

    default:

        grade = 'F';

    }

    cout << "The grade is: " << grade << endl;

    return 0;

}

### Output



## Q3

#include <iostream>

#include <vector>

using namespace std;

double add(const vector<double>& operands);

double subtract(const vector<double>& operands);

double multiply(const vector<double>& operands);

double divide(const vector<double>& operands);

int main() {

    int numOperands;

    char operation;

    vector<double> operands;

    cout << "Enter the operation you want to perform (+, -, \*, /): ";

    cin >> operation;

    // Validate operation

    while (operation != '+' && operation != '-' && operation != '\*' && operation != '/') {

        cout << "Invalid operation. Try again\n";

    }

    cout << "How many operands do you want to use? ";

    cin >> numOperands;

    if (numOperands < 2) {

        cout << "You need at least two operands to perform an operation.\n";

        return 1;

    }

    cout << "Enter the operands:\n";

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

        double value;

        cin >> value;

        operands.push\_back(value);

    }

    double result;

    switch (operation) {

    case '+':

        result = add(operands);

        break;

    case '-':

        result = subtract(operands);

        break;

    case '\*':

        result = multiply(operands);

        break;

    case '/':

        result = divide(operands);

        break;

    default:

        cout << "Unexpected error.\n";

        return 1;

    }

    cout << "The result is: " << result << endl;

    return 0;

}

double add(const vector<double>& operands) {

    double sum = 0;

    for (double op : operands) {

        sum += op;

    }

    return sum;

}

double subtract(const vector<double>& operands) {

    double result = operands[0];

    for (size\_t i = 1; i < operands.size(); ++i) {

        result -= operands[i];

    }

    return result;

}

double multiply(const vector<double>& operands) {

    double product = 1;

    for (double op : operands) {

        product \*= op;

    }

    return product;

}

double divide(const vector<double>& operands) {

    double result = operands[0];

    for (size\_t i = 1; i < operands.size(); ++i) {

        if (operands[i] == 0) {

            cout << "Error: Division by zero detected.\n";

            exit(1);

        }

        result /= operands[i];

    }

    return result;

}

### Output

A black background with white text

Description automatically generated

## Q4

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

    int matrix[3][3]{};

    cout << "Enter 9 integers for the 3x3 matrix:\n";

    // Input data into the matrix using nested for loops

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 3; j++) {

            cout << "Enter element [" << i << "][" << j << "]: ";

            cin >> matrix[i][j];

        }

    }

    // Output the 3x3 matrix

    cout << "\nThe 3x3 matrix is:\n";

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 3; j++) {

            cout << setw(5) << left << matrix[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

### Output

A computer screen shot of a number

Description automatically generated

## Q5

#include <iostream>

using namespace std;

int main() {

    int rows = 6;

    for (int i = 1; i <= rows; i++) {

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

            cout << "\*";

        }

        cout << endl;

    }

    return 0;

}

### Output

A black background with white stars

Description automatically generated

## Q6

// This program creates a 6x7 matrix and stores the calender in it, then it prints it out.

#include <iostream>

#include <iomanip>

#include <vector>

#include <string>

int main()

{

    // Initialize a 6x7 matrix to represent the calendar (6 weeks, 7 days each)

    std::vector<std::vector<int>> calendar(6, std::vector<int>(7, 0));

    // August 2014 starts on a Friday

    int start\_day = 5; // 0 for Sunday, so Friday is 5

    int days\_in\_month = 31;

    int day = 1; // Start from the 1st of August

    // Fill the calendar with the days of August 2014

    for (int index = start\_day; index < start\_day + days\_in\_month; ++index)

    {

        int week = index / 7;

        int day\_of\_week = index % 7;

        calendar[week][day\_of\_week] = day;

        ++day;

    }

    // Weekday labels

    std::vector<std::string> weekdays = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};

    // Print the calendar

    std::cout << "Calendar of August 2014:" << std::endl;

    for (const auto &weekday : weekdays)

    {

        std::cout << weekday << "\t";

    }

    std::cout << std::endl;

    for (const auto &row : calendar)

    {

        for (const auto &col : row)

        {

            if (col != 0)

            {

                std::cout << std::setw(3) << col << "\t";

            }

            else

            {

                std::cout << "   \t";

            }

        }

        std::cout << std::endl;

    }

    return 0;

}

### Output

A screenshot of a calendar

Description automatically generated

## Q7

#include <iostream>

#include <cstdlib>

using namespace std;

int main()

{

    int number;

    // Input the 6-digit number

    cout << "Enter a 6-digit number: ";

    cin >> number;

    // Check if the number is a 6-digit number

    if (number < 100000 || number > 999999)

    {

        cout << "The number entered is not a 6-digit number." << endl;

        return 1;

    }

    int reversedNumber = 0, sumOfDigits = 0;

    int originalNumber = number;

    // Reverse the number and calculate the sum of its digits

    while (number > 0)

    {

        int digit = number % 10;

        reversedNumber = reversedNumber \* 10 + digit;

        sumOfDigits += digit;

        number /= 10;

    }

    // Output the results

    cout << "Reversed Number: " << reversedNumber << endl;

    cout << "Sum of Digits: " << sumOfDigits << endl;

    return 0;

}

### Output

A black background with white text

Description automatically generated

# Set 2

## Q1

#include <iostream>

using namespace std;

long factorial(int number);

long pow(int base, int exponent);

static double SumofSequence(int x, int count);

int main()

{

    cout << "Enter a number \n";

    int number = 0;

    int count = 0;

    cin >> number;

    cout << "Enter number of terms for the sequence\n";

    cin >> count;

    cout << "sum" << SumofSequence(number, count) << "\n";

    return 0;

}

static double SumofSequence(int x, int count)

{

    double sum = x;

    count = count \* 2 - 1;

    bool plusOrMinus = true;

    for (int i = 3; i <= count; i += 2)

    {

        if (plusOrMinus)

        {

            sum += static\_cast<double>(pow(x, i)) / factorial(i - 1);

        }

        else

        {

            sum -= static\_cast<double>(pow(x, i)) / factorial(i - 1);

        }

        plusOrMinus = !plusOrMinus;

    }

    return sum;

}

long pow(int base, int exponent)

{

    // returns the power of a number

    long result = 1;

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

    {

        result \*= base;

    }

    return result;

}

long factorial(int number)

{

    // returns the factorial of a number

    long result = 1;

    for (int i = 1; i <= number; i++)

    {

        result \*= i;

    }

    return result;

}

### Output

A black background with white text

Description automatically generated

## Q2

#include <iostream>

using namespace std;

char \*CreateExpression(int line, int pos);

int main()

{

    for (int i = 1; i < 6; i++)

    {

        static char line[30];

        int offset = 15 - 3 \* i;

        for (int j = 0; j < offset; j++)

        {

            line[j] = ' ';

        }

        for (int j = 0; j < i; j++)

        {

            char \*expression = CreateExpression(i, j + 1);

            for (int k = 0; k < 5; k++)

            {

                line[j \* 6 + k + offset] = expression[k];

            }

            line[j \* 6 + 5 + offset] = ' ';

        }

        line[6 \* i - 1 + offset] = '\0';

        cout << line;

        cout << "\n";

    }

    return 0;

}

char \*CreateExpression(int line, int pos)

{

    char expression[5] = {'x', '^', line + '0', '+', pos + '0'};

    return expression;

}

### Output

A black background with white numbers and symbols

Description automatically generated

## Q3

#include <iostream>

using namespace std;

int numUpto = 5;

void MakePattern(char \*pattern, int line);

void align(char \*pattern, int line);

int main()

{

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

    {

        char \*pattern = new char[numUpto + i + 2];

        align(pattern, i);

        MakePattern(pattern, i);

        pattern[numUpto + i + 1] = '\0';

        cout << pattern << endl;

        delete[] pattern;

    }

    return 0;

}

void MakePattern(char \*pattern, int line)

{

    for (int i = 0; i <= line; i++)

    {

        pattern[numUpto + i] = '0' + i;

        pattern[numUpto - i] = '0' + i;

    }

}

void align(char \*pattern, int line)

{

    for (int i = 0; i < numUpto - line; i++)

    {

        pattern[i] = ' ';

    }

}

### Output

A black background with white numbers

Description automatically generated

## Q4

#include <iostream>

#include <cstdlib>

using namespace std;

int main()

{

    char num1[7];

    char num2[7];

    char num3[7];

    char num1rev[7] = "000000";

    char num2rev[7] = "000000";

    char num3rev[7] = "000000";

    cout << "Enter first number: ";

    cin >> num1;

    cout << "Enter second number: ";

    cin >> num2;

    cout << "Enter third number: ";

    cin >> num3;

    // Reverse the numbers

    for (int i = 0; i < 6; i++)

    {

        num1rev[i] = num1[5 - i];

        num2rev[i] = num2[5 - i];

        num3rev[i] = num3[5 - i];

    }

    // Convert the numbers to integers

    int num1Int = atoi(num1rev);

    int num2Int = atoi(num2rev);

    int num3Int = atoi(num3rev);

    // Print the reversed numbers

    cout << "Reversed first number: ";

    cout << num1rev << endl;

    cout << "Reversed second number: ";

    cout << num2rev << endl;

    cout << "Reversed third number: ";

    cout << num3rev << endl;

    // Finding the largest of the reversed

    int largest = num1Int;

    if (num2Int > largest)

        largest = num2Int;

    if (num3Int > largest)

        largest = num3Int;

    cout << "Largest number among the reversed numbers: " << largest << endl;

    return 0;

}

### Output

A screen shot of numbers

Description automatically generated

## Q5

#include <iostream>

void generateFibonacci(int \*fibArray, int n)

{

    if (n <= 0)

        return;

    if (n >= 1)

        fibArray[0] = 0;

    if (n >= 2)

        fibArray[1] = 1;

    for (int i = 2; i < n; ++i)

    {

        fibArray[i] = fibArray[i - 1] + fibArray[i - 2];

    }

}

static unsigned long factorial(int n)

{

    unsigned long result = 1;

    for (int i = 1; i <= n; i++)

    {

        result \*= i;

    }

    return result;

}

int main()

{

    int n;

    std::cout << "Enter the number of Fibonacci numbers to generate: ";

    std::cin >> n;

    if (n <= 0)

    {

        std::cout << "Invalid input. Please enter a positive integer." << std::endl;

        return 1;

    }

    int \*fibArray = new int[n];

    generateFibonacci(fibArray, n);

    std::cout << "Num\tFact\n";

    for (int i = 0; i < n; ++i)

    {

        std::cout << fibArray[i] << "\t" << factorial(fibArray[i]) << "\n";

    }

    std::cout << std::endl;

    delete[] fibArray;

    return 0;

}

### Output

A screen shot of a computer

Description automatically generated

## Q6

#include <iostream>

using namespace std;

int main()

{

    char string[1000];

    int stringLength = 0;

    cout << "Enter a string: ";

    cin >> string;

    for (stringLength = 0; string[stringLength] != '\0'; stringLength++)

    {

        continue;

    }

    bool \*isVowel = new bool[stringLength];

    bool \*isNotChar = new bool[stringLength];

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

    {

        if (string[i] == 'a' || string[i] == 'e' || string[i] == 'i' || string[i] == 'o' || string[i] == 'u')

        {

            isVowel[i] = true;

        }

        else

        {

            isVowel[i] = false;

        }

        if ((string[i] >= 'A' && string[i] <= 'Z') || (string[i] >= 'a' && string[i] <= 'z'))

        {

            isNotChar[i] = false;

        }

        else

        {

            isNotChar[i] = true;

        }

    }

    cout << "Vowels are at positions: ";

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

    {

        if (isVowel[i] == true)

        {

            cout << i + 1 << ", ";

        }

    }

    cout << "\nNon-Characters are at positions: ";

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

    {

        if (isNotChar[i] == true)

        {

            cout << i + 1 << ", ";

        }

    }

    int firstVowel = -1;

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

    {

        if (isVowel[i] == true)

        {

            firstVowel = i + 1;

            break;

        }

    }

    cout << "\nfirst vowel is at position: " << firstVowel;

    return 0;

}

### Output

A black screen with white text

Description automatically generated

## Q7

#include <iostream>

#include <cstring>

using namespace std;

void removeConsecutiveCharacters(const char \*input, char \*output, int &originalCount, int &processedCount)

{

    originalCount = strlen(input);

    int j = 0;

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

    {

        if (i == 0 || input[i] != input[i - 1])

        {

            output[j++] = input[i];

        }

    }

    output[j] = '\0';

    processedCount = j;

}

int main()

{

    char input[1000];

    char output[1000];

    cout << "Enter a string: ";

    cin.getline(input, 1000);

    int originalCount = 0;

    int processedCount = 0;

    removeConsecutiveCharacters(input, output, originalCount, processedCount);

    cout << "Original string: " << input << endl;

    cout << "Original character count: " << originalCount << endl;

    cout << "Processed string: " << output << endl;

    cout << "Processed character count: " << processedCount << endl;

    return 0;

}

### Output

A black background with white text

Description automatically generated