qn_1_pr.cpp > ...

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main(int argc, char *argv[])
{

    int numberOfTerms;
    if (argc == 1)
    {
        cout << "Command line input not passed!" << endl<<" Please Enter the number
        of terms ";
        cin >> numberOfTerms;
    }
    else
    {
        numberOfTerms = stoi(argv[1]);
    }
    cout << "Entered number of terms : " << numberOfTerms << endl;
    float sumOfSeries = 0;
    for (int i = 1; i <= numberOfTerms; i++)
    {
        sumOfSeries += pow(-1, i + 1) / pow(i, i);
    }
    cout << "Sum of the series till " << numberOfTerms << " terms is " << sumOfSeries;
    return 0;
```

qn_2_pr.cpp > ...

```cpp
#include <iostream>
using namespace std;
int main()
{
    int arr[5] = {1, 1, 2, 3, 2};
    int uni[5] = {0, 0, 0, 0, 0};
    uni[0] = arr[0];
    int index = 1;
    bool found;
    for (int i = 1; i < 5; i++)
    {
        found = false;
        for (int j = 0; j < i; j++)
        {
            if (arr[i] == arr[j])
            {
                found = true;
                break;
            }
        }
        if (!found)
        {
            uni[index] = arr[i];
            index++;
        }
    }
    for (int i = 0; i < 5; i++)
    {
        cout<< uni[i] << " ";
    }
    return 0;
}
```

qn_3_pr.cpp > main(int, char * [])

```cpp
#include <iostream>
#include <string>
using namespace std;

int noOfChar(string str, char ch) {
int count = 0;
for (int i = 0; i < str.length(); i++)
{
    if (ch == str[i])
    {
        count++;
    }
}
return count;
}
int main(int argc, char *argv[])
{
    string text = argv[1];
    string printedChar;
    cout << "String : " << text << endl;
    cout << "| char |occurance |" << endl;
    for (int i = 0; i < text.length(); i++)
    {
        printedChar += text[i];
        if (noOfChar(printedChar, text[i]) == 1)
        {
            cout << "| " << text[i] << " | " << noOfChar(text, text[i]) << " | "<<endl; }
        }
}
```

qn_4_pr.cpp > main()

```cpp
#include <iostream>
#include <string>
using namespace std;

void showAddress(string); // a string concatenate(string, string); //b void compare(string, string); //c int stringLength(string); //d string uppercase(string); //e string reverse(string); //f string insertString (string, string, int);//g
int main()
{
    char key;
    while (key != ' ')
    {
    cout << "String Manipulation Program : Press a-g to manipulate strings, press spacebar to exit "<<endl;
    cout<<" a.Show address of each character in string "<< endl;
    cout << "b.Concatenate two strings."<<endl;

    cout<< " c. Compare two strings "<< endl;
    cout<<"d.Calculate length of the string(use pointers) "<<endl;
    cout << "e. Convert all lowercase characters to uppercase " << endl;
    cout << "f.Reverse the string "<<endl;
    cout<<"g. Insert a string in another string at a user specified position" << endl;
        char response;
        cout << "Enter your response : ";
        cin >> response;
        switch (response)
        {
            case 'a':
            {
                string str;
                cout << "Enter a string : ";
                cin >> str;
                showAddress(str);
```

```
30              showAddress(str);
31              break;
32          }
33          case 'b':
34          {
35              string str1, str2;
36              cout << "Enter first string : ";
37              cin.ignore();
38              getline(cin, str1);
39              cout << "Enter second string : ";
40              getline(cin, str2);
41              string concinated = concatenate(str1, str2);
42              cout << concinated << endl;
43              break;
44          }
45          case 'c':
46          {
47              string str1, str2;
48              cout << "Enter first string : ";
49              cin.ignore();
50              getline(cin, str1);
51              cout << "Enter second string : ";
52              getline(cin, str2);
53              compare(str1, str2);
54              break;
55          }
56          case 'd':
57          {
58              string str;
59              cout << "Enter a string : ";
60              cin >> str;
61              int len = stringLength(str);
```

```
61          int len = stringLength(str);
62          cout << len << endl;
63          break;
64      }
65      case 'e':
66      {
67          string str;
68          cout << "Enter a string : ";
69          cin >> str;
70          string upper_str = uppercase(str);
71          cout << upper_str << endl;
72          break;
73      }
74      case 'f':
75      {
76          string str;
77          cout << "Enter a string : ";
78          cin >> str;
79          string reversed_str = reverse(str);
80          cout << reversed_str << endl;
81          break;
82      }
83      case 'g':
84      {
85          string str1, str2;
86          int pos;
87          cout << "Enter first string 1 : ";
88          cin.ignore();
89          getline(cin, str1);
90          cout << "Enter second string 2 : ";
91          getline(cin, str2);
92          cout << "Enter position where you want to insert string 2: ";
```

```
93              cin >>pos;
94              string newStr = insertString(str1, str2, pos);
95              cout << newStr << endl;
96              break;
97          }
98          default:
99          {
100             break;
101         }
102         }
103     }
104     }
105     void showAddress(string str)
106     {
107         for (int i = 0; i < str.length(); i++)
108         {
109             cout << "Position of " << str[i] << ": " << (void *)str[i] << endl;
110         }
111     }
112     string concatenate(string str1, string str2)
113     {
114         string conc;
115         conc = str1 + str2;
116         return conc;
117     }
118     void compare(string str1, string str2)
119     {
120         if (str1 > str2)
121         {
122             cout << str1 << " > " << str2 << endl;
123         }
124         else if (str1 < str2)
```

```
124         else if (str1 < str2)
125         {
126             cout << str2 << " < " << str1 << endl;
127         }
128         else
129         {
130             cout << str1 << " = " << str2 << endl;
131         }
132     }
133     int len(string & x)
134     {
135         int count = 0;
136         for (int i : x)
137         {
138             count++;
139         }
140         return (count);
141     }
142     int stringLength(string str)
143     {
144         int count = 0;
145         for (int i = 0; i < str.length(); i++)
146         {
147             count++;
148         }
149         return count;
150     }
151     string uppercase(string str)
152     {
153         string str_upper;
154         for (int i = 0; i < str.length(); i++)
155         {
```

```
155         {
156             char letter = str[i];
157             str_upper += toupper(letter);
158         }
159         return (str_upper);
160     }
161     string reverse(string str)
162     {
163         string reversed_str;
164         for (int i = 0; i < str.length(); i++)
165         {
166             char letter = str[i];
167             reversed_str = letter + reversed_str;
168         }
169         return (reversed_str);
170     }
171     string insertString(string str1, string str2, int pos)
172     {
173         string newStr;
174         for (int i = 0; i < pos; i++)
175         {
176             newStr += str1[i];
177         }
178         newStr += str2;
179         for (int i = pos; i < str1.length(); i++)
180         {
181             newStr += str1[i];
182         }
183         return (newStr);
184     }
```

qn_5_pr.cpp > displayArray(int [], int)

```cpp
#include <iostream>
using namespace std;

void displayArray(int newarr[], int len);
int main()
{
    int arr1[] = {1, 2, 3, 15, 65};
    int arr2[] = {0, 11, 12, 14};
    int len1 = sizeof(arr1) / sizeof(int);
    int len2 = sizeof(arr2) / sizeof(int);
    int newarr[len1 + len2];
    for (int i = 0; i < len1; i++)
    {
        newarr[i] = arr1[i];
    }
    for (int i = 0; i < len2; i++)
    {
        newarr[i + len1] = arr2[i];
    }
    int n = sizeof(newarr) / sizeof(int);
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (newarr[j] > newarr[j + 1])
            {
                int temp = newarr[j];
                newarr[j] = newarr[j + 1];
                newarr[j + 1] = temp;
            }
        }
    }
```

```
33      cout << "Orded Array 1 : " << endl;
34      displayArray(arr1, len1);
35      cout << "Orded Array 2 : " << endl;
36      displayArray(arr2, len2);
37      cout << "Orded Merged Array : " << endl;
38      displayArray(newarr, n);
39  }
40  void displayArray(int newarr[], int len)
41  {
42      for (int i = 0; i < len; i++)
43      {
44          cout << newarr[i] << " ";
45      }
46      cout << endl;
47  }
```

qn_6_pr.cpp > ...

```cpp
#include <iostream>
using namespace std;
int main()
{
    int set[] = {1, 2, 3, 5, 81, 7, 8, 9};
    int size = sizeof(set) / sizeof(int);
    int search_int;
    cout << "Enter number to be searched : ";
    cin >> search_int;
    bool found = false;
    int pos;
    for (int i = 0; i < size; i++)
    {
        if (search_int == set[i])
        {
            found = true;
            pos = i;
            break;
        }
    }
    if (found)
    {
        cout << search_int << " found at " << pos + 1 << " position";
    }
    else
    {
        cout << search_int << " is not in the set";
    }
}
```

qn_7_pr.cpp > main()

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout << "Enter num 1 : ";
    cin >> a;
    cout << "Enter num 2 : ";
    cin >> b;
    if (a < b)
    {
        while (b % a != 0)
        {
            a = b % a;
        }
        cout << "Required GCD : " << a;
    }
    else
    {
        while (a % b != 0)
        {
            b = a % b;
        }
        cout << "Required GCD : " << b;
    }
}
```

qn_8_pr.cpp > Matrix > arr

```cpp
/* 8. Create a Matrix class. Write a menu-driven program to perform following Matrix operations (exceptions should be thrown by the functions if matrices passed to them are incompatible and handled by the main() function): a. Sum b. Product c. Transpose
 */
#include <iostream>
#include <vector>
using namespace std;
class Matrix
{
    int row, col;
    vector<vector<int>> arr;

public:
    Matrix() {}
    Matrix(int noOfRow, int noOfCol) : row(noOfRow), col(noOfCol), arr(
                                                    noOfRow, vector<int>(noOfCol, 0)) {}
    void inputMatrix()
    {
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < col; j++)
            {
                int element;
                cout << "Enter element at (" << i << "," << j << ")position : ";
                cin >>element;
                arr[i][j] = element;
            }
        }
    }
    void displayMatrix() const
    {
        for (int i = 0; i < row; i++)
```

```
        {
            for (int j = 0; j < col; j++)
            {
                cout << arr[i][j] << " ";
            }
            cout << endl;
        }
    }
    int getElement(int i, int j) const
    {
        return arr[i][j];
    }
    void setElement(int i, int j, int ele)
    {
        arr[i][j] = ele;
    }
    friend Matrix sum(const Matrix &, const Matrix &);
    friend Matrix product(const Matrix &, const Matrix &);
    friend Matrix transpose(const Matrix &);
};
Matrix sum(const Matrix &A, const Matrix &B)
{
    int row = A.row;
    int col = A.col;
    Matrix C(row, col);
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            C.setElement(i, j, A.getElement(i, j) + B.getElement(i, j));
        }
    }
```

```
63      return C;
64  }
65  Matrix product(const Matrix &A, const Matrix &B)
66  {
67      int rowA = A.row;
68      int colA = A.col;
69      int rowB = B.row;
70      int colB = B.col;
71      if (colA != rowB)
72      {
73          throw "Matrix dimensions are not compatible for multiplication!";
74      }
75      Matrix C(rowA, colB);
76      for (int i = 0; i < rowA; i++)
77      {
78          for (int j = 0; j < colB; j++)
79          {
80              int sum = 0;
81              for (int k = 0; k < colA; k++)
82              {
83                  sum += A.getElement(i, k) * B.getElement(k, j);
84              }
85              C.setElement(i, j, sum);
86          }
87      }
88      return C;
89  }
90  Matrix transpose(const Matrix &A)
91  {
92      int row = A.row;
93      int col = A.col;
```

```
94      Matrix C(col, row);
95      for (int i = 0; i < col; i++)
96      {
97          for (int j = 0; j < row; j++)
98          {
99              C.setElement(i, j, A.getElement(j, i));
100         }
101     }
102     return C;
103 }
104 int main()
105 {
106     int rows, cols;
107     cout << "Enter the number of rows in the matrices: ";
108     cin >> rows;
109     cout << "Enter the number of columns in the matrices: ";
110     cin >> cols;
111     Matrix A(rows, cols);
112     Matrix B(rows, cols);
113     cout << "Enter the elements of the first matrix:" << endl;
114     A.inputMatrix();
115     cout << "Enter the elements of the second matrix:" << endl;
116     B.inputMatrix();
117     int choice;
118     cout << "Select an operation:" << endl;
119     cout << "1. Sum" << endl;
120     cout << "2. Product" << endl;
121     cout << "3. Transpose" << endl;
122     cout << "Enter your choice (1-3): ";
123     cin >> choice;
124     Matrix result;
125     try
```

```
126     {
127         switch (choice)
128         {
129         case 1:
130             result = sum(A, B);
131             cout << "Sum of the matrices:" << endl;
132             result.displayMatrix();
133             break;
134         case 2:
135             result = product(A, B);
136             cout << "Product of the matrices:" << endl;
137             result.displayMatrix();
138             break;
139         case 3:
140             result = transpose(A);
141             cout << "Transpose of the matrix:" << endl;
142             result.displayMatrix();
143             break;
144         default:
145             cout << "Invalid choice!" << endl;
146         }
147     }
148     catch (const char *errorMessage)
149     {
150         cout << "Error: " << errorMessage << endl;
151     }
152     return 0;
153 }
```

qn_9_pr.cpp > Person > display(void)

```cpp
#include<iostream>
using namespace std;

class Person
{
protected:
    string name;

public:
    Person(string Pname)
    {
        name = Pname;
    }
    virtual void display(void)
    {
        cout << "Name : " << name << endl;
    }
};
class Student : public Person
{
    string course;
    float marks;
    int year;

public:
    Student(string name, string Mcourse, float mark, int yrs) : Person(
                                                                    name)
    {
        course = Mcourse;
        marks = mark;
        year = yrs;
    }
    void display(void)
    {
        cout << "Name : " << name << endl;
        cout << "Course : " << course << endl;
        cout << "Marks : " << marks << endl;
        cout << "Year : " << year << endl;
    }
```

```
    }
};
class Employee : public Person
{
    string department;
    float salary;

public:
    Employee(string Ename, string dept, float sal) : Person(Ename)
    {
        department = dept;
        salary = sal;
    }
    void display(void)
    {
        cout << "Name : " << name << endl;
        cout << "Department : " << department << endl;
        cout << "Salary : " << salary << endl;
    }
};
int main()
{
    Person *perPtr;
    Student S1("Ravi", "CS", 123, 2023);
    Employee E1("Anshu", "Tech", 900000);
    perPtr = &E1;
    cout << "Employee's details :" << endl;
    perPtr->display();
    perPtr = &S1;
    cout << endl;
    cout << "Student's details :" << endl;
    perPtr->display();
    return 0;
}
```

qn_10_pr.cpp > main()

```cpp
#include <iostream>
#include <cmath>
#include <cstring>
using namespace std;
class Error
{
    int err_code;
    string err_desc;

public:
    Error(int c, string errMsg)
    {
        err_code = c;
        err_desc = errMsg;
    }
    void err_display(void)
    {
        cout << "Error Code: " << err_code << endl
             << "Error Description: " << err_desc << endl;
    }
};
class Triangle
{
    float side1, side2, side3;

public:
    Triangle() {}
    Triangle(float a, float b, float c)
    {
        try
        {
            if (a <= 0 || b <= 0 || c <= 0)
            {
                throw Error(001, "Sides cannot be negative or 0!");
            }
            if (a >= b + c || b >= a + c || c >= a + b)
            {
                throw Error(002, "Either of side exceeds the sum of other two sides!");
            }
            side1 = a;
            side2 = b;
            side3 = c;
        }
        catch (Error e)
        {
            e.err_display();
        }
    }
```

```
    }
    float area()
    {
        float s = (side1 + side2 + side3) / 2;
        float area = sqrt(s * (s - side1) * (s - side2) * (s - side3));
        return area;
    }
    float area(float base, float height)
    {
        try
        {
            float area = (base * height) / 2;
            if (area == 0)
            {
                throw Error(003, "Invalid Base or Height of Right triangle ");
            }
            return area;
        }
        catch (Error e)
        {
            e.err_display();
        }
    }
};

int main()
{
    Triangle DEF(0, 3, 4);
    Triangle ABC(3, 4, 5);
    float area = ABC.area();
    cout << "Area of general Trianle ABC is " << area << endl;
    Triangle PQR;
    float rArea = PQR.area(4, 6);
    cout << "Area of Right angled Trianle ABC is " << rArea << endl;
}
```

qn_11_pr.cpp > main()

```cpp
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream file("textfile.txt");
    ofstream fileCopy("copiedfile.txt");
    string line;
    while (file)
    {
        getline(file, line);
        string copyLine;
        for (int i = 0; i < line.length(); i++)
        {
            if (line[i] != ' ')
            {
                copyLine += line[i];
            }
        }
        fileCopy << copyLine << endl;
    }
}
```