Assignment 1

Write a program in C++ to implement a Stack using class.

Stack.cpp ...

```
#include <iostream>

const int SIZE = 5;

class Stack {
private:
   int top;
   int stk[SIZE];

   int isEmpty() { return top == -1; }
   int isFull() { return top == SIZE-1; }

public:
   // constructor
   Stack() { top = -1; }

   // pushes one element onto the stack
   void push(int data) {
      if(isFull())
         std::cout << "Stack Overflow\n";
      else {
         top++;
         stk[top] = data;
         std::cout << "Pushed " << data << " onto stack.\n";
      }
   }

   // removes one element from the top of the stack
   void pop() {
      if(isEmpty())
         std::cout << "Stack underflow\n";

      else {
         std::cout << "Poped: " << stk[top] << "\n";
         top--;
      }
   }

   // display the stack
   void display() {
      if(!isEmpty()) {
         std::cout << "Stack elements are:\n";
         for(int i = top; i >= 0; i--)
            std::cout << stk[i] << "\n";
      }
```

```
        else
            std::cout << "Stack is Empty\n";

    }
};

int main()
{
    Stack obj;

    std::cout << "\tMenu:\n"
              << "1. Push\n"
              << "2. Pop\n"
              << "3. Display\n"
              << "4. Exit\n";

    while(1) {
        int choice;
        std::cout << "Enter your choice between 1 to 4.\n";
        std::cin >> choice;

        switch(choice) {
            case 1:
                std::cout << "Enter the data\n";
                int data;
                std::cin >> data;
                obj.push(data);

                break;

            case 2:
                obj.pop();
                break;

            case 3:
                obj.display();
                break;

            case 4:
                std::cout << "Terminating…\n";
                exit(0);
                break;

            default:
                std::cout << "Invalid choice.\nPlease enter a number between 1 to 4.\n";
                break;

        }
    }

    return 0;
}
```

Output ...

```
    Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice between 1 to 4.
1
Enter the data
4
Pushed 4 onto stack.
Enter your choice between 1 to 4.
1
Enter the data
6
Pushed 6 onto stack.
Enter your choice between 1 to 4.
1
Enter the data
8
Pushed 8 onto stack.
Enter your choice between 1 to 4.
1
Enter the data
35
Pushed 35 onto stack.
Enter your choice between 1 to 4.
1
Enter the data
26
Pushed 26 onto stack.
Enter your choice between 1 to 4.
1
Enter the data
37
Stack Overflow
Enter your choice between 1 to 4.
2
Poped: 26
Enter your choice between 1 to 4.
3
Stack elements are:
35
8
6
4
Enter your choice between 1 to 4.
2
Poped: 35
Enter your choice between 1 to 4.
2
Poped: 8
Enter your choice between 1 to 4.
2
Poped: 6
Enter your choice between 1 to 4.
2
Poped: 4
Enter your choice between 1 to 4.
2
Stack underflow
Enter your choice between 1 to 4.
4
Terminating…
```

Assignment 2

Write a program in C++ to print the following pattern using class structure when

1. The height of the pattern is given.
2. The middle character of the pattern is given.

```
ABCDEDCBA
ABCD  DCBA
ABC    CBA
AB      BA
A        A
```

Pattern.cpp ...

```
#include <iostream>

class Pattern
{
private:
   std::string s;

   // returns the first  line of the pattern given 'n'
   // e.g. n = 5 return ABCDEDCBA
   std::string buildFirst(int n) {
      std::string res;

      int A = 65;
      for(int i = -n; i < n-1; i++) {
         if(i < 0)
            res += (char) (A + n+i);
         else
            res += (char) (A + n-i-2);
      }

      return res;
   }

   // repeats space character for n number of times
   std::string repeat(int n) {
      if(n > 0) {
         std::string res;
         for(int i = 0; i < n; i++) res += " ";
         return res;
      }
      else return "";
   }

public:
   void setChar(char ch) {
      int diff = (int) (ch - 'A') + 1;
      s = buildFirst(diff);
   }
```

```
    void setLen(int len) {
        s = buildFirst(len);
    }

    void print() {
        std::cout << "\n";
        int len = s.length()/2;
        for (int i = 0; i <= len; i++) {
            int start = len - i;
            int end = len + i;
            int diff = end-start+1;

            std::cout << s << "\n";
            s.replace(s.begin()+start, s.begin()+end+1, repeat(diff));
        }
        std::cout << "\n";
    }
};

int main() {
    Pattern p;

    std::cout << "\n\t\tMENU\n"
              << "1. Print pattern with char.\n"
              << "2. Print pattern with length.\n"
              << "3. Exit.\n";

    while(1) {
        std::cout << "Enter your choice between 1 to 3.\n";
        int choice;
        std::cin >> choice;

        switch(choice) {
            case 1:
                std::cout << "Enter the character.\n";
                char charInp;
                std::cin >> charInp;

                p.setChar(toupper(charInp));
                p.print();

            break;

            case 2:
                std::cout << "Enter the length.\n";
                int len;
                std::cin >> len;

                p.setLen(len);
                p.print();

            break;

            case 3:
                std::cout << "Terminating the program\n";
```

```
            exit(0);
         break;

         default:
            std::cout << "Please enter a number between 1 to 3.\n";
         break;
      }
   }
}
```

Output ...

```
            MENU
1. Print pattern with char.
2. Print pattern with length.
3. Exit.
Enter your choice between 1 to 3.
1
Enter the character.
M

ABCDEFGHIJKLMLKJIHGFEDCBA
ABCDEFGHIJKL    LKJIHGFEDCBA
ABCDEFGHIJK       KJIHGFEDCBA
ABCDEFGHIJ          JIHGFEDCBA
ABCDEFGHI            IHGFEDCBA
ABCDEFGH              HGFEDCBA
ABCDEFG                 GFEDCBA
ABCDEF                    FEDCBA
ABCDE                      EDCBA
ABCD                        DCBA
ABC                          CBA
AB                            BA
A                              A

Enter your choice between 1 to 3.
2
Enter the length.
8

ABCDEFGHGFEDCBA
ABCDEFG   GFEDCBA
ABCDEF       FEDCBA
ABCDE          EDCBA
ABCD             DCBA
ABC                CBA
AB                   BA
A                      A
Enter your choice between 1 to 3.
3
Terminating the program
```

Assignment 3

Construct a class AREA with requisite data members. Write a C++ program using appropriate member functions to calculate the area of the rectangle, scalene triangle and circle using function overloading.

Area.cpp ...

```
#include <iostream>
#include <cmath>

class AREA {
public:
   // Rectangle area calculation
   void calculateArea(double length, double width) {
      double area = length * width;
      std::cout << "Area of Rectangle: " << area << std::endl;
   }

   // Scalene triangle area calculation
   void calculateArea(double A, double B, double C) {
      double S = (A + B + C) / 2; // semi-perimeter
      double area = sqrt(S * (S-A) * (S-B) * (S-C));
      std::cout << "Area of Scalene Triangle: " << area << std::endl;
   }

   // Circle area calculation
   void calculateArea(double radius) {
      const double PI = 22/7.0;
      double area = PI * radius * radius;
      std::cout << "Area of Circle: " << area << std::endl;
   }
};

int main() {
   AREA shape;
   int choice;
   double length, width, radius;
   double sideA, sideB, sideC;

   do {
      std::cout << "===== MENU =====" << std::endl;
      std::cout << "1. Calculate Area of Rectangle" << std::endl;
      std::cout << "2. Calculate Area of Scalene Triangle" << std::endl;
      std::cout << "3. Calculate Area of Circle" << std::endl;
      std::cout << "4. Exit" << std::endl;
      std::cout << "Enter your choice: ";
      std::cin >> choice;

      switch (choice) {
         case 1:
            std::cout << "Enter length and width of the rectangle: ";
            std::cin >> length >> width;
            shape.calculateArea(length, width);
```

```
            break;
        case 2:
            std::cout << "Enter the three sides of the scalene triangle: ";
            std::cin >> sideA >> sideB >> sideC;
            shape.calculateArea(sideA, sideB, sideC);
            break;
        case 3:
            std::cout << "Enter radius of the circle: ";
            std::cin >> radius;
            shape.calculateArea(radius);
            break;
        case 4:
            std::cout << "Exiting the program." << std::endl;
            break;
        default:
            std::cout << "Invalid choice. Please try again." << std::endl;
            break;
    }
    std::cout << std::endl;
  } while (choice != 4);

  return 0;
}
```

Output ...

```
===== MENU =====
1. Calculate Area of Rectangle
2. Calculate Area of Scalene Triangle
3. Calculate Area of Circle
4. Exit
Enter your choice: 1
Enter length and width of the rectangle: 25
10
Area of Rectangle: 250

===== MENU =====
1. Calculate Area of Rectangle
2. Calculate Area of Scalene Triangle
3. Calculate Area of Circle
4. Exit
Enter your choice: 2
Enter the three sides of the scalene triangle: 8 6 10
Area of Scalene Triangle: 24

===== MENU =====
1. Calculate Area of Rectangle
2. Calculate Area of Scalene Triangle
3. Calculate Area of Circle
4. Exit
Enter your choice: 3
Enter radius of the circle: 11
Area of Circle: 380.286

===== MENU =====
1. Calculate Area of Rectangle
2. Calculate Area of Scalene Triangle
3. Calculate Area of Circle
4. Exit
Enter your choice: 4
Exiting the program.
```

Assignment 4

Write a C++ program that replaces two or more consecutive blanks in a string by a single blank.

RemoveSpace.cpp ...

```
#include <iostream>
#include <cstring>

class MyString {
private:
   char* str;

public:
   MyString(const char* str) {
      int len = std::strlen(str);
      this->str = new char[len + 1];
      std::strcpy(this->str, str);
   }

   ~MyString() {
      delete[] str;
   }

   char* modify();
};

char* MyString::modify() {
   int len = std::strlen(str);
   char* res = new char[len + 1];
   int index = 0;

   for (int i = 0; i < len; i++) {
      if (!(str[i] == ' ' && str[i + 1] == ' ')) {
         res[index++] = str[i];
      }
   }

   res[index] = '\0';

   return res;
}

int main() {
   char str[100];
   std::cout << "Enter your string.\n";
   std::cin.getline(str, 100);
   MyString s(str);
   char* modifiedStr = s.modify();
   std::cout << "Modified string:\n" << modifiedStr << "\n";
   delete[] modifiedStr;
   return 0;
}
```

Output ...

```
Enter your string.
I          love          my          country.
Modified string:
I love my country.
```

Assignment 5

Construct a class TIME having three data members of hour, minute, seconds and some member functions. One constructor should initialize these data members to 0 and another should initialize those to fixed values passed from main function.

Write a member function to display time in the format HH:MM:SS and another member function to add two TIME objects and hence develop a program in C++ to add two given TIME object using operator overloading.

Time.cpp ...

```
#include <iostream>

class TIME {
private:
   int hour;
   int minute;
   int second;


public:
   // Default constructor
   TIME() : hour(0), minute(0), second(0) {}

   // Parameterized constructor
   TIME(int h, int m, int s) : hour(h), minute(m), second(s) {}

   // Member function to display time in HH:MM:SS format
   void displayTime() const {
      printf("%02d:%02d:%02d\n", this->hour, this->minute, this->second);
   }

   // Operator overloading to add two TIME objects
   TIME operator+(const TIME& other) const {
      TIME sum;
      sum.hour = hour + other.hour;
      sum.minute = minute + other.minute;
      sum.second = second + other.second;

      if (sum.second >= 60) {
         sum.minute += sum.second / 60;
         sum.second %= 60;
      }

      if (sum.minute >= 60) {
         sum.hour += sum.minute / 60;
         sum.minute %= 60;
      }

      return sum;
   }
```

```
};

int main() {
   int h1, m1, s1, h2, m2, s2;

   std::cout << "Enter time 1 (hours minutes seconds): ";
   std::cin >> h1 >> m1 >> s1;

   std::cout << "Enter time 2 (hours minutes seconds): ";
   std::cin >> h2 >> m2 >> s2;

   TIME t1(h1, m1, s1);
   TIME t2(h2, m2, s2);

   std::cout << "Time 1: ";
   t1.displayTime();

   std::cout << "Time 2: ";
   t2.displayTime();

   TIME sum = t1 + t2;
   std::cout << "Sum of Time 1 and Time 2: ";
   sum.displayTime();

   return 0;
}
```

Output ...

```
Enter time 1 (hours minutes seconds): 2 10 20
Enter time 2 (hours minutes seconds): 3 50 40
Time 1: 02:10:20
Time 2: 03:50:40
Sum of Time 1 and Time 2: 06:01:00
```

Assignment 6

Write program to add, subtract and multiply two matrices using class MATRIX. The class should contain a method to read row and column of a matrix, and read the matrix elements, a method to display a matrix. The class also contain 3 different methods for addition, subtraction and multiplication. Write appropriate main function to implement class.

Matrix1.cpp ...

```
#include <iostream>
#include <iomanip>

using namespace std;

const int MAX_SIZE = 10;

class Matrix {
private:
   int rows, cols;
   double matrix[MAX_SIZE][MAX_SIZE];

public:
   void readRow();
   void readCol();
   void readMatrix();
   void display();
   Matrix add(const Matrix other);
   Matrix subtract(const Matrix other);
   Matrix multiply(const Matrix other);
};



int main() {
   Matrix A, B;

   cout << "Matrix1:\n";
   A.readRow();
   A.readCol();
   A.readMatrix();
   A.display();

   cout << "\nMatrix2:\n";
   B.readRow();
   B.readCol();
   B.readMatrix();
   B.display();

   cout << "\nMenu\n"
      << "Select an operation.\n"
      << "1. Add two matrices\n"
      << "2. Subtract two matrices.\n"
      << "3. Multiply two matrices.\n"
      << "4. Exit.\n";
```

```
   while(1) {
      int choice;
      cout << "Enter your choice.\n";
      cin >> choice;

      switch (choice)
      {
      case 1: {
         cout << "A + B =\n";
         A.add(B).display();
         break;
      }
      case 2:{
         cout << "A - B =\n";
         A.subtract(B).display();
         break;
      }
      case 3:{
         cout << "A x B =\n";
         A.multiply(B).display();
         break;
      }
      case 4:
         cout << "Terminating the program\n";
         exit(0);
         break;

      default:
         cout << "Invalid choice.\n";
         break;
      }
   }
   return 0;
}



// Implementation of Matrix class

void Matrix::readRow()
{
   cout << "Enter the number of rows: ";
   cin >> this->rows;
}

void Matrix::readCol()
{
   cout << "Enter the number of columns: ";
   cin >> this->cols;
}

void Matrix::readMatrix()
{
   cout << "Enter the matrix elements:\n";
   for(int i = 0; i < rows; ++i) {
```

```
      for(int j = 0; j < cols; ++j) {
         cout << "Element [" << i << "," << j << "]: ";
         cin >> matrix[i][j];
      }
   }
}

void Matrix::display()
{
   cout << "Matrix:\n";
   for(int i = 0; i < rows; ++i) {
      for(int j = 0; j < cols; ++j) {
         cout << setw(4) << matrix[i][j];
      }
      cout << "\n";
   }
   cout << "\n";
}

Matrix Matrix::add(Matrix other) {
   if((rows != other.rows) || (cols != other.cols)) {
      cout << "Dimension mismatch.\nMatrix addition is not possible.\n";
      return *this;
   }

   Matrix result;
   result.rows = rows;
   result.cols = cols;

   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         result.matrix[i][j] = matrix[i][j] + other.matrix[i][j];
      }
   }

   return result;
}

Matrix Matrix::subtract(Matrix other) {
   if((rows != other.rows) || (cols != other.cols)) {
      cout << "Dimension mismatch.\nMatrix subtraction is not possible.\n";
      return *this;
   }

   Matrix result;
   result.rows = rows;
   result.cols = cols;

   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         result.matrix[i][j] = matrix[i][j] - other.matrix[i][j];
      }
   }

   return result;
```

```
}

Matrix Matrix::multiply(Matrix other) {
   if(other.rows != cols) {
      cout << "Dimension mismatch.\nMatrix addition is not possible.\n";
      return *this;
   }

   Matrix result;
   result.rows = rows;
   result.cols = other.cols;

   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < other.cols; j++) {
         result.matrix[i][j] = 0;
         for (int k = 0; k < cols; k++) {
            result.matrix[i][j] += matrix[i][k] * other.matrix[k][j];
         }
      }
   }

   return result;
}
```

Output ...

```
Matrix1:
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix elements:
Element [0,0]: 1
Element [0,1]: 3
Element [0,2]: 6
Element [1,0]: 3
Element [1,1]: 8
Element [1,2]: 3
Element [2,0]: 5
Element [2,1]: 4
Element [2,2]: 8
Matrix:
  1   3   6
  3   8   3
  5   4   8


Matrix2:
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix elements:
Element [0,0]: 8
Element [0,1]: 2
Element [0,2]: 7
Element [1,0]: 3
Element [1,1]: 9
Element [1,2]: 3
```

```
Element [2,0]: 0
Element [2,1]: 5
Element [2,2]: 2
Matrix:
  8   2   7
  3   9   3
  0   5   2


Menu
Select an operation.
1. Add two matrices
2. Subtract two matrices.
3. Multiply two matrices.
4. Exit.
Enter your choice.
1
A + B =
Matrix:
  9   5  13
  6  17   6
  5   9  10

Enter your choice.
2
A - B =
Matrix:
 -7   1  -1
  0  -1   0
  5  -1   6

Enter your choice.
3
A x B =
Matrix:
 17  59  28
 48  93  51
 52  86  63

Enter your choice.
4
Terminating the program
```

Assignment 7

Write program to add, subtract and multiply two matrices using class MATRIX. The class should contain a method to read row and column of a matrix, and read the matrix elements, a method to display a matrix. Use operator overloading to perform addition, subtraction and multiplication. Write appropriate main function to implement class.

Matrix2.cpp ...

```
#include <iostream>
#include <iomanip>

using namespace std;

const int MAX_SIZE = 10;

class Matrix {
private:
   int rows, cols;
   double matrix[MAX_SIZE][MAX_SIZE];

public:
   void readRow();
   void readCol();
   void readMatrix();
   void display();
   Matrix operator+(const Matrix other);
   Matrix operator-(const Matrix other);
   Matrix operator*(const Matrix other);
};


int main() {
   Matrix A, B, result;

   cout << "Matrix1:\n";
   A.readRow();
   A.readCol();
   A.readMatrix();
   A.display();

   cout << "\nMatrix2:\n";
   B.readRow();
   B.readCol();
   B.readMatrix();
   B.display();

   cout << "\nMenu\n"
      << "Select an operation.\n"
      << "1. Add two matrices\n"
      << "2. Subtract two matrices.\n"
      << "3. Multiply two matrices.\n"
      << "4. Exit.\n";

   while(1) {
```

```
        int choice;
        cout << "Enter your choice.\n";
        cin >> choice;

        switch (choice)
        {
        case 1: {
           cout << "A + B =\n";
           result = A + B;
           result.display();
           break;
        }
        case 2:{
           cout << "A - B =\n";
           result = A - B;
           result.display();
           break;
        }
        case 3:{
           cout << "A x B =\n";
           result = A * B;
           result.display();
           break;
        }
        case 4:
           cout << "Terminating the program\n";
           exit(0);
           break;

        default:
           cout << "Invalid choice.\n";
           break;
        }
     }
     return 0;
}



// Implementation of Matrix class

void Matrix::readRow()
{
   cout << "Enter the number of rows: ";
   cin >> this->rows;
}

void Matrix::readCol()
{
   cout << "Enter the number of columns: ";
   cin >> this->cols;
}

void Matrix::readMatrix()
{
```

```
    cout << "Enter the matrix elements:\n";
    for(int i = 0; i < rows; ++i) {
        for(int j = 0; j < cols; ++j) {
            cout << "Element [" << i << "," << j << "]: ";
            cin >> matrix[i][j];
        }
    }
}

void Matrix::display()
{
    cout << "Matrix:\n";
    for(int i = 0; i < rows; ++i) {
        for(int j = 0; j < cols; ++j) {
            cout << setw(4) << matrix[i][j];
        }
        cout << "\n";
    }
    cout << "\n";
}

Matrix Matrix::operator+(Matrix other) {
    if((rows != other.rows) || (cols != other.cols)) {
        cout << "Dimension mismatch.\nMatrix addition is not possible.\n";
        return *this;
    }

    Matrix result;
    result.rows = rows;
    result.cols = cols;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result.matrix[i][j] = matrix[i][j] + other.matrix[i][j];
        }
    }

    return result;
}

Matrix Matrix::operator-(Matrix other) {
    if((rows != other.rows) || (cols != other.cols)) {
        cout << "Dimension mismatch.\nMatrix subtraction is not possible.\n";
        return *this;
    }

    Matrix result;
    result.rows = rows;
    result.cols = cols;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result.matrix[i][j] = matrix[i][j] - other.matrix[i][j];
        }
    }
```

```
    return result;
}

Matrix Matrix::operator*(Matrix other) {
    if(other.rows != cols) {
        cout << "Dimension mismatch.\nMatrix addition is not possible.\n";
        return *this;
    }

    Matrix result;
    result.rows = rows;
    result.cols = other.cols;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < other.cols; j++) {
            result.matrix[i][j] = 0;
            for (int k = 0; k < cols; k++) {
                result.matrix[i][j] += matrix[i][k] * other.matrix[k][j];
            }
        }
    }

    return result;
}
```

Output ...

```
Matrix1:
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix elements:
Element [0,0]: 1
Element [0,1]: 3
Element [0,2]: 6
Element [1,0]: 3
Element [1,1]: 8
Element [1,2]: 3
Element [2,0]: 5
Element [2,1]: 4
Element [2,2]: 8
Matrix:
  1   3   6
  3   8   3
  5   4   8


Matrix2:
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix elements:
Element [0,0]: 8
Element [0,1]: 2
Element [0,2]: 7
Element [1,0]: 3
```

```
Element [1,1]: 9
Element [1,2]: 3
Element [2,0]: 0
Element [2,1]: 5
Element [2,2]: 2
Matrix:
  8  2  7
  3  9  3
  0  5  2


Menu
Select an operation.
1. Add two matrices
2. Subtract two matrices.
3. Multiply two matrices.
4. Exit.
Enter your choice.
1
A + B =
Matrix:
  9  5 13
  6 17  6
  5  9 10

Enter your choice.
2
A - B =
Matrix:
 -7  1 -1
  0 -1  0
  5 -1  6

Enter your choice.
3
A x B =
Matrix:
 17 59 28
 48 93 51
 52 86 63

Enter your choice.
4
Terminating the program
```

Assignment 8

Write program to add, subtract and multiply two matrices using class MATRIX. The class should contain a method to read row and column of a matrix, and read the matrix elements, a method to display a matrix. Use friend function to implement addition, subtraction and multiplication. Write appropriate main function to implement class.

Matrix3.cpp ...

```
#include <iostream>
#include <iomanip>

using namespace std;

const int MAX_SIZE = 10;

class Matrix {
private:
   int rows, cols;
   double matrix[MAX_SIZE][MAX_SIZE];

public:
   void readRow();
   void readCol();
   void readMatrix();
   void display();
   friend Matrix add(const Matrix A, const Matrix B);
   friend Matrix subtract(const Matrix A, const Matrix B);
   friend Matrix multiply(const Matrix A, const Matrix B);

};


int main() {
   Matrix A, B, result;

   cout << "Matrix1:\n";
   A.readRow();
   A.readCol();
   A.readMatrix();
   A.display();

   cout << "\nMatrix2:\n";
   B.readRow();
   B.readCol();
   B.readMatrix();
   B.display();

   cout << "\nMenu\n"
      << "Select an operation.\n"
      << "1. Add two matrices\n"
      << "2. Subtract two matrices.\n"
      << "3. Multiply two matrices.\n"
      << "4. Exit.\n";
```

```
   while(1) {
      int choice;
      cout << "Enter your choice.\n";
      cin >> choice;

      switch (choice)
      {
      case 1: {
         cout << "A + B =\n";
         result = add(A, B);
         result.display();
         break;
      }
      case 2:{
         cout << "A - B =\n";
         result = subtract(A, B);
         result.display();
         break;
      }
      case 3:{
         cout << "A x B =\n";
         result = multiply(A, B);
         result.display();
         break;
      }
      case 4:
         cout << "Terminating the program\n";
         exit(0);
         break;

      default:
         cout << "Invalid choice.\n";
         break;
      }
   }
   return 0;
}



// Implementation of Matrix class

void Matrix::readRow()
{
   cout << "Enter the number of rows: ";
   cin >> this->rows;
}

void Matrix::readCol()
{
   cout << "Enter the number of columns: ";
   cin >> this->cols;
}

void Matrix::readMatrix()
```

```
{
   cout << "Enter the matrix elements:\n";
   for(int i = 0; i < rows; ++i) {
      for(int j = 0; j < cols; ++j) {
         cout << "Element [" << i << "," << j << "]: ";
         cin >> matrix[i][j];
      }
   }
}

void Matrix::display()
{
   cout << "Matrix:\n";
   for(int i = 0; i < rows; ++i) {
      for(int j = 0; j < cols; ++j) {
         cout << setw(4) << matrix[i][j];
      }
      cout << "\n";
   }
   cout << "\n";
}


Matrix add(const Matrix A, const Matrix B)
{
   if((A.rows != B.rows) || (A.cols != B.cols)) {
      cout << "Dimension mismatch.\nMatrix addition is not possible.\n";
      return A;
   }

   Matrix result;
   result.rows = A.rows;
   result.cols = B.cols;

   for (int i = 0; i < A.rows; i++) {
      for (int j = 0; j < B.cols; j++) {
         result.matrix[i][j] = A.matrix[i][j] + B.matrix[i][j];
      }
   }

   return result;
}

Matrix subtract(const Matrix A, const Matrix B)
{
   if((A.rows != B.rows) || (A.cols != B.cols)) {
      cout << "Dimension mismatch.\nMatrix subtraction is not possible.\n";
      return A;
   }

   Matrix result;
   result.rows = A.rows;
   result.cols = A.cols;

   for (int i = 0; i < A.rows; i++) {
```

```
      for (int j = 0; j < A.cols; j++) {
         result.matrix[i][j] = A.matrix[i][j] - B.matrix[i][j];
      }
   }

   return result;

}

Matrix multiply(const Matrix A, const Matrix B)
{
   if(B.rows != A.cols) {
      cout << "Dimension mismatch.\nMatrix addition is not possible.\n";
      return A;
   }

   Matrix result;
   result.rows = A.rows;
   result.cols = B.cols;

   for (int i = 0; i < A.rows; i++) {
      for (int j = 0; j < B.cols; j++) {
         result.matrix[i][j] = 0;
         for (int k = 0; k < A.cols; k++) {
            result.matrix[i][j] += A.matrix[i][k] * B.matrix[k][j];
         }
      }
   }

   return result;
}
```

Output ...

```
Matrix1:
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix elements:
Element [0,0]: 1
Element [0,1]: 3
Element [0,2]: 6
Element [1,0]: 3
Element [1,1]: 8
Element [1,2]: 3
Element [2,0]: 5
Element [2,1]: 4
Element [2,2]: 8
Matrix:
  1   3   6
  3   8   3
  5   4   8


Matrix2:
Enter the number of rows: 3
```

```
Enter the number of columns: 3
Enter the matrix elements:
Element [0,0]: 8
Element [0,1]: 2
Element [0,2]: 7
Element [1,0]: 3
Element [1,1]: 9
Element [1,2]: 3
Element [2,0]: 0
Element [2,1]: 5
Element [2,2]: 2
Matrix:
  8   2   7
  3   9   3
  0   5   2


Menu
Select an operation.
1. Add two matrices
2. Subtract two matrices.
3. Multiply two matrices.
4. Exit.
Enter your choice.
1
A + B =
Matrix:
  9   5  13
  6  17   6
  5   9  10

Enter your choice.
2
A - B =
Matrix:
 -7   1  -1
  0  -1   0
  5  -1   6

Enter your choice.
3
A x B =
Matrix:
 17  59  28
 48  93  51
 52  86  63

Enter your choice.
4
Terminating the program
```

Assignment 9

Write program using dynamic initialization, appropriate constructors to add, subtract and multiply two matrices using class MATRIX. The class should contain a method to display a matrix. The class also contain 3 different methods for addition, subtraction and multiplication. Write appropriate main function to implement class.

Matrix4.cpp ...

```
#include <iostream>
#include <iomanip>

using namespace std;

class Matrix {
private:
   int row, col;
   double **matrix;

public:
   Matrix(int row, int col);
   Matrix(const Matrix& copy);
   ~Matrix();

   void fill();
   void display();

   Matrix add(const Matrix &other);
   Matrix subtract(const Matrix& other);
   Matrix multiply(const Matrix& other);
};


Matrix init() {
   int row, col;
   cout << "Enter the rows and columns: \n";
   cin >> row >> col;

   Matrix matrix(row, col);
   cout << "Enter the elements (row-wise):\n";
   matrix.fill();
   return matrix;
}


int main() {
   cout << "Menu\n"
        << "Select an operation.\n"
        << "1. Add two matrices\n"
        << "2. Subtract two matrices.\n"
        << "3. Multiply two matrices.\n"
        << "4. Exit.\n";

   while(1) {
      int choice;
      cout << "Enter your choice.\n";
```

```
cin >> choice;

switch (choice)
{
case 1: {
   cout << "Enter details for Matrix A:\n";
   Matrix A = init();
   A.display();

   cout << "Enter details for Matrix B:\n";
   Matrix B = init();
   B.display();

   cout << "A + B =\n";
   A.add(B).display();

   break;
}
case 2:{
   cout << "Enter details for Matrix A:\n";
   Matrix A = init();
   A.display();

   cout << "Enter details for Matrix B:\n";
   Matrix B = init();
   B.display();

   cout << "A - B =\n";
   A.subtract(B).display();

   break;
}
case 3:{
   cout << "Enter details for Matrix A:\n";
   Matrix A = init();
   A.display();

   cout << "Enter details for Matrix B:\n";
   Matrix B = init();
   B.display();

   cout << "A x B =\n";
   A.multiply(B).display();

   break;
}
case 4:
   cout << "Terminating the program\n";
   exit(0);
   break;

default:
   cout << "Invalid choice.\n";
   break;
}
```

```
    }

    return 0;
}



// Implementation of Matrix class

Matrix::Matrix(int row, int col) : row(row), col(col)
{
    // Allocate memory for the matrix
    matrix = new double*[row];
    for (int i = 0; i < row; ++i) {
        matrix[i] = new double[col]();
    }
}

Matrix::Matrix(const Matrix& copy) : row(copy.row), col(copy.col)
{
    matrix = new double*[row];
    for (int i = 0; i < row; ++i) {
        matrix[i] = new double[col];
        for (int j = 0; j < col; ++j) {
            matrix[i][j] = copy.matrix[i][j];
        }
    }
}

Matrix::~Matrix()
{
    // Deallocate memory for the matrix
    for (int i = 0; i < row; ++i) {
        delete[] matrix[i];
    }
    delete[] matrix;
}


Matrix Matrix::add(const Matrix& other)
{
    if((row != other.row) || (col != other.col)) {
        cout << "Matrix addition is not possible.\n"
             << "Row and columns of both matrices must be same.\n";
        return Matrix(0, 0);
    }

    else {
        Matrix res(other);

        for(int i = 0; i < row; ++i) {
            for(int j = 0; j < col; ++j) {
                res.matrix[i][j] += matrix[i][j];
            }
```

```
      }

      return res;
   }
}

Matrix Matrix::subtract(const Matrix& other)
{
   if((row != other.row) || (col != other.col)) {
      cout << "Matrix subtraction is not possible.\n"
          << "Row and columns of both matrices must be same.\n";
      return Matrix(0, 0);
   }

   else {
      Matrix res(other);

      for(int i = 0; i < row; ++i) {
         for(int j = 0; j < col; ++j) {
            res.matrix[i][j] -= matrix[i][j];
         }
      }

      return res;
   }
}

Matrix Matrix::multiply(const Matrix& other)
{
   if(other.row != col) {
      cout << "Matrix multiplication is not possible.\n"
          << "Column of the first matrix must be equal to row of the second matrix.\n";
      return Matrix(0, 0);
   }

   else {
      Matrix res(row, other.col);

      for(int i = 0; i < res.row; ++i) {
         for(int j = 0; j < res.col; ++j) {
            for(int k = 0; k < col; ++k) {
               res.matrix[i][j] += matrix[i][k] * other.matrix[k][j];
            }
         }
      }

      return res;
   }
}


void Matrix::fill()
{
   for(int i = 0; i < row; ++i) {
      for(int j = 0; j < col; ++j) {
```

```
            cout << "Enter the [" << i << "," << j << "] th element: ";
            cin >> matrix[i][j];
        }
    }
}

void Matrix::display()
{
    cout << "Matrix:\n";
    for(int i = 0; i < row; ++i) {
        for(int j = 0; j < col; ++j) {
            cout << setw(3) << matrix[i][j];
        }
        cout << "\n";
    }
    cout << "\n";
}
```

Output ...

```
Menu
Select an operation.
1. Add two matrices
2. Subtract two matrices.
3. Multiply two matrices.
4. Exit.
Enter your choice.
1
Enter details for Matrix A:
Enter the rows and columns:
2
3
Enter the elements (row-wise):
Enter the [0,0] th element: 46
Enter the [0,1] th element: 27
Enter the [0,2] th element: 37
Enter the [1,0] th element: 3
Enter the [1,1] th element: 76
Enter the [1,2] th element: 4
Matrix:
 46 27 37
  3 76  4

Enter details for Matrix B:
Enter the rows and columns:
2
3
Enter the elements (row-wise):
Enter the [0,0] th element: 56
Enter the [0,1] th element: 3
Enter the [0,2] th element: 3
Enter the [1,0] th element: 7
Enter the [1,1] th element: 2
Enter the [1,2] th element: 8
```

```
Matrix:
 56  3  3
  7  2  8

A + B =
Matrix:
102 30 40
 10 78 12

Enter your choice.
3
Enter details for Matrix A:
Enter the rows and columns:
3
2
Enter the elements (row-wise):
Enter the [0,0] th element: 1
Enter the [0,1] th element: 5
Enter the [1,0] th element: 7
Enter the [1,1] th element: 3
Enter the [2,0] th element: 8
Enter the [2,1] th element: 2
Matrix:
  1  5
  7  3
  8  2

Enter details for Matrix B:
Enter the rows and columns:
2
3
Enter the elements (row-wise):
Enter the [0,0] th element: 7
Enter the [0,1] th element: 8
Enter the [0,2] th element: 2
Enter the [1,0] th element: 1
Enter the [1,1] th element: 6
Enter the [1,2] th element: 4
Matrix:
  7  8  2
  1  6  4

A x B =
Matrix:
 12 38 22
 52 74 26
 58 76 24

Enter your choice.
4
Terminating the program
```

Assignment 10

Write a program in C++ using class to allow the statement S1 += S2 using += operator;
where S2 us added (concatenated) to S1 and the result left in S1.
The operator should also permit the result of the operation to be used in other calculation,
as in S3 = S1 += S2

CustomString.cpp ...

```
#include <iostream>
#include <cstring>

class StringConcatenator {
private:
   char str[100];

public:
   // Default constructor
   StringConcatenator() {
         str[0] = '\0';
   }

   // Parameterized constructor
   StringConcatenator(const char* s) {
         std::strcpy(str, s);
   }

   // Concatenate another string to the current string
   StringConcatenator& operator+= (const StringConcatenator& other) {
      std::strcat(str, other.str);
      return *this;
   }

   // Getter method for the string value
   const char* getString() const {
         return str;
   }
};

int main() {
   char input1[100], input2[100];

   std::cout << "Enter string 1: ";
   std::cin.getline(input1, sizeof(input1));

   std::cout << "Enter string 2: ";
   std::cin.getline(input2, sizeof(input2));
```

```
    StringConcatenator S1(input1);
    StringConcatenator S2(input2);
    StringConcatenator S3;

    S1 += S2;
    S3 = S1 += S2;

    std::cout << "S1: " << S1.getString() << std::endl;
    std::cout << "S2: " << S2.getString() << std::endl;
    std::cout << "S3: " << S3.getString() << std::endl;

    return 0;
}
```

Output ...

```
Enter string 1: Hello, my name is
Enter string 2: Mouli Dutta.
S1: Hello, my name is Mouli Dutta.Mouli Dutta.
S2: Mouli Dutta.
S3: Hello, my name is Mouli Dutta.Mouli Dutta.
```

Assignment 11

Write a program in C++ to compare and concatenate two strings.

CustomString.cpp ...

```
#include <iostream>
#include <cstring>

class StringManipulator {
private:
   char str[100];

public:
   // Default constructor
   StringManipulator() {
      str[0] = '\0';  // Initialize the string with an empty C-style string
   }

   // Parameterized constructor
   StringManipulator(const char* s) {
      std::strcpy(str, s);
   }

   // Compare two strings
   int compare(const char* other) const {
      return std::strcmp(str, other);
   }

   // Concatenate another string to the current string
   void concatenate(const char* other) {
      std::strcat(str, other);
   }

   // Getter method for the string value
   const char* getString() const {
      return str;
   }
};

int main() {
   char input1[100], input2[100];

   std::cout << "Enter string 1: ";
   std::cin.getline(input1, sizeof(input1));

   std::cout << "Enter string 2: ";
   std::cin.getline(input2, sizeof(input2));

   StringManipulator str1(input1);
```

```
    StringManipulator str2(input2);

    int comparisonResult = str1.compare(str2.getString());
    if (comparisonResult < 0) {
        std::cout << "String 1 is less than String 2." << std::endl;
    } else if (comparisonResult > 0) {
        std::cout << "String 1 is greater than String 2." << std::endl;
    } else {
        std::cout << "String 1 is equal to String 2." << std::endl;
    }

    str1.concatenate(str2.getString());
    std::cout << "Concatenated string: " << str1.getString() << std::endl;

    return 0;
}
```

Output ...

```
Enter string 1: I am Mouli
Enter string 2: Hello World
String 1 is greater than String 2.
Concatenated string: I am MouliHello World
```

Assignment 12

Write a program in C++ to compare and concatenate two strings using dynamic initialization, constructor and operator overloading.

CustomString.cpp ...

```
#include <iostream>
#include <cstring>

class String {
private:
   char* str;

public:
   // Default constructor
   String() : str(nullptr) {}

   // Constructor with dynamic initialization
   String(const char* s) {
      int length = std::strlen(s);
      str = new char[length + 1];
      std::strcpy(str, s);
   }

   // Destructor to release dynamically allocated memory
   ~String() {
      delete[] str;
   }

   // Compare two strings
   int compare(const String& other) const {
      return std::strcmp(str, other.str);
   }

   // Concatenate two strings
   String operator+(const String& other) const {
      int length1 = std::strlen(str);
      int length2 = std::strlen(other.str);
      char* concatenatedString = new char[length1 + length2 + 1];
      std::strcpy(concatenatedString, str);
      std::strcat(concatenatedString, other.str);
      return String(concatenatedString);
   }

   // Getter method for the string value
   const char* getString() const {
      return str;
   }
};

int main() {
   char input1[100], input2[100];
```

```
    // Input string 1
    std::cout << "Enter string 1: ";
    std::cin.getline(input1, sizeof(input1));

    // Input string 2
    std::cout << "Enter string 2: ";
    std::cin.getline(input2, sizeof(input2));

    // Create objects and perform operations
    String S1(input1);
    String S2(input2);
    String S3 = S1 + S2;

    // Compare the two strings
    int comparisonResult = S1.compare(S2);

    // Display the comparison result
    if (comparisonResult == 0) {
        std::cout << "The strings are equal." << std::endl;
    } else if (comparisonResult < 0) {
        std::cout << "String 1 is less than string 2." << std::endl;
    } else {
        std::cout << "String 1 is greater than string 2." << std::endl;
    }

    // Display the concatenated string
    std::cout << "Concatenated string: " << S3.getString() << std::endl;

    return 0;
}
```

Output ...

```
Enter string 1: Hello
Enter string 2: World
String 1 is less than string 2.
Concatenated string: HelloWorld
```

Assignment 13

In an office all the staffs get Basic and HRA but managers get additional allowance. The office has branches in Kolkata, Delhi and Darjeeling. In Kolkata office all the staffs get special allowance, in Delhi city allowance and in Darjeeling hill allowance.

Write a C++ program to get data and show the data for each branch.

Office.cpp ...

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

class Staff {
private:
   int isManager() {
      cout << "Enter 1 if the staff is a manager otherwise 0.\n";
      int isManager;
      cin >> isManager;
      return isManager;
   }
protected:
   string name;
   double basic;
   double hra;
   double additionalAllowence;

public:
   void getData() {
      cout << "Enter staff name: ";
      cin.ignore();
      getline(cin, name);
      cout << "Enter basic salary: ";
      cin >> basic;
      cout << "Enter HRA: ";
      cin >> hra;

      if(isManager() == 1) {
         cout << "Enter the additional allowence: ";
         cin >> additionalAllowence;
      } else {
         additionalAllowence = 0;
      }
   }

   void showData() {
      cout << "Staff name: " << name << endl;
      cout << "Basic salary: " << basic << endl;
      cout << "HRA: " << hra << endl;
      cout << "Additional Allowence: " << additionalAllowence << endl;
   }
};

class Kolkata : public Staff {
```

```
private:
   int specialAllowance;
public:
   void getData() {
      Staff::getData();
      cout << "Enter special allowance: ";
      cin >> specialAllowance;
   }
   void showData() {
      Staff::showData();
      cout << "Special allowance: " << specialAllowance << endl;
   }
};

class Delhi : public Staff {
private:
   int cityAllowance;
public:
   void getData() {
      Staff::getData();
      cout << "Enter city allowance: ";
      cin >> cityAllowance;
   }
   void showData() {
      Staff::showData();
      cout << "City allowance: " << cityAllowance << endl;
   }
};

class Darjeeling : public Staff {
private:
   int hillAllowance;
public:
   void getData() {
      Staff::getData();
      cout << "Enter hill allowance: ";
      cin >> hillAllowance;
   }
   void showData() {
      Staff::showData();
      cout << "Hill allowance: " << hillAllowance << endl;
   }
};

int main() {
   int choice;
   Kolkata kolkata;
   Delhi delhi;
   Darjeeling darjeeling;

   do {
      cout << "========== MENU ==========\n"
           << "1. Enter data for Kolkata branch\n"
           << "2. Enter data for Delhi branch\n"
           << "3. Enter data for Darjeeling branch\n"
```

```
            << "4. Display data for Kolkata branch\n"
            << "5. Display data for Delhi branch\n"
            << "6. Display data for Darjeeling branch\n"
            << "0. Exit\n";

        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter data for Kolkata branch:" << endl;
                kolkata.getData();
                break;
            case 2:
                cout << "Enter data for Delhi branch:" << endl;
                delhi.getData();
                break;
            case 3:
                cout << "Enter data for Darjeeling branch:" << endl;
                darjeeling.getData();
                break;
            case 4:
                cout << "Data for Kolkata branch:" << endl;
                kolkata.showData();
                break;
            case 5:
                cout << "Data for Delhi branch:" << endl;
                delhi.showData();
                break;
            case 6:
                cout << "Data for Darjeeling branch:" << endl;
                darjeeling.showData();
                break;
            case 0:
                cout << "Exiting program." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }

        cout << "\n";

    } while (choice != 0);

    return 0;
}
```

Output ...

```
========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
```

```
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 1
Enter data for Kolkata branch:
Enter staff name: Mouli Dutta
Enter basic salary: 42000
Enter HRA: 5000
Enter 1 if the staff is a manager otherwise 0.
1
Enter the additional allowence: 2000
Enter special allowance: 6000

========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 4
Data for Kolkata branch:
Staff name: Mouli Dutta
Basic salary: 42000
HRA: 5000
Additional Allowence: 2000
Special allowance: 6000

========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 2
Enter data for Delhi branch:
Enter staff name: John
Enter basic salary: 30000
Enter HRA: 6000
Enter 1 if the staff is a manager otherwise 0.
0
Enter city allowance: 8000

========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 5
Data for Delhi branch:
```

```
Staff name: John
Basic salary: 30000
HRA: 6000
Additional Allowence: 0
City allowance: 8000

========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 3
Enter data for Darjeeling branch:
Enter staff name: Cindy
Enter basic salary: 60000
Enter HRA: 20000
Enter 1 if the staff is a manager otherwise 0.
1
Enter the additional allowence: 10000
Enter hill allowance: 3500

========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 6
Data for Darjeeling branch:
Staff name: Cindy
Basic salary: 60000
HRA: 20000
Additional Allowence: 10000
Hill allowance: 3500

========== MENU ==========
1. Enter data for Kolkata branch
2. Enter data for Delhi branch
3. Enter data for Darjeeling branch
4. Display data for Kolkata branch
5. Display data for Delhi branch
6. Display data for Darjeeling branch
0. Exit
Enter your choice: 0
Exiting program.
```

Assignment 14

Write a C++ program using class to read a text file and replace two or more consecutive blanks with a single blank and first letter of each word in uppercase to another file.

File.cpp ...

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

class TextFileProcessor {
private:
   string inputFileName;
   string outputFileName;

public:
   TextFileProcessor(const string& inputFileName, const string& outputFileName)
      : inputFileName(inputFileName), outputFileName(outputFileName) {}

   void processFile() {
      ifstream inputFile(inputFileName);
      ofstream outputFile(outputFileName);

      if (!inputFile) {
         cout << "Error opening input file." << endl;
         return;
      }

      if (!outputFile) {
         cout << "Error creating output file." << endl;
         return;
      }

      char prevChar = '\0';
      char currentChar;
      bool isFirstLetter = true;

      while (inputFile.get(currentChar)) {
         if (isspace(currentChar)) {
            if (prevChar != ' ') {
               outputFile << ' ';
            }
         } else {
            if (isFirstLetter) {
               outputFile << static_cast<char>(toupper(currentChar));
               isFirstLetter = false;
            } else {
               outputFile << currentChar;
            }
         }

         prevChar = currentChar;
```

```
            if (currentChar == '\n') {
                isFirstLetter = true;
            }
        }

        inputFile.close();
        outputFile.close();

        cout << "File processing complete." << endl;
    }
};

int main() {
    string inputFileName, outputFileName;

    cout << "Enter input file name: ";
    cin >> inputFileName;

    cout << "Enter output file name: ";
    cin >> outputFileName;

    TextFileProcessor processor(inputFileName, outputFileName);
    processor.processFile();

    return 0;
}
```

Output ...

```
Enter input file name: InputTest.txt
Enter output file name: OutputTest.txt
File processing complete.
```

InputTest.txt

```
hello    world!
I        am          Mouli Dutta. how are    you?
this is a    test.
```

OutputTest.txt

```
Hello world! I am Mouli Dutta. how are you? This is a test.
```