|  |  |
| --- | --- |
| **Name** | **Taaha Hussain Khan** |
| **Roll No:** | **L1F21BSCS0917** |
| **Section** | **D12** |

**Task 01**

//Taaha Hussain Khan

//L1F21BSCS0917

//D12

#include <iostream>

using namespace std;

void fibonacci(int n) {

    int t1 = 0, t2 = 1, nextTerm = 0;

    cout << "Fibonacci Series: ";

    cout << t1 << " " << t2 << " ";

    while (nextTerm <= n) {

        nextTerm = t1 + t2;

        if (nextTerm <= n)

            cout << nextTerm << " ";

        t1 = t2;

        t2 = nextTerm;

    }

    cout << endl;

}

int fib(int n) {

    if (n <= 1)

        return n;

    return fib(n-1) + fib(n-2);

}

int main() {

    int num;

    cout << "Enter a number: ";

    cin >> num;

    fibonacci(num);

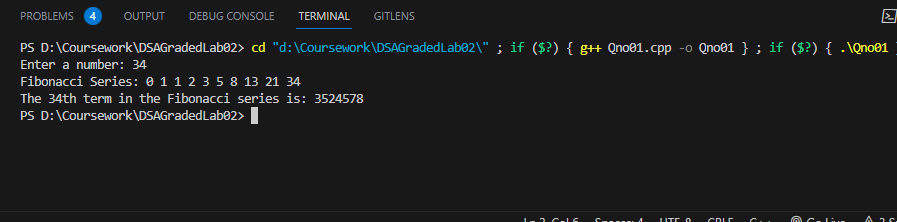
    int nthTerm = fib(num-1);

    cout << "The " << num << "th term in the Fibonacci series is: " << nthTerm << endl;

    return 0;

}

**Output:**



**Task02**

//Taaha Hussain Khan

//L1F21BSCS0917

//D12

#include <iostream>

using namespace std;

// Recursive function

int gcd(int a, int b) {

    if (b == 0)

        return a;

    return gcd(b, a % b);

}

int main() {

    int a, b;

    cout << "Enter two numbers: ";

    cin >> a >> b;

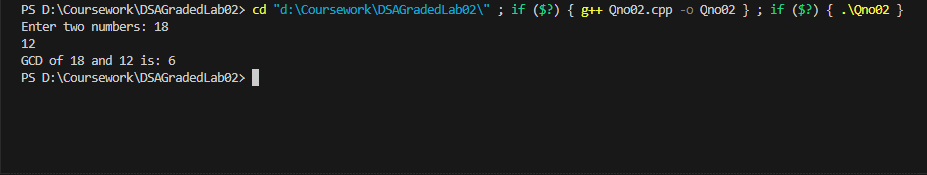
    int result = gcd(a, b);

    cout << "GCD of " << a << " and " << b << " is: " << result << endl;

    return 0;

}

**Output:**



**Task03**

//Taaha Hussain Khan

//L1F21BSCS0917

//D12

#include <iostream>

using namespace std;

class CircularQueue {

private:

    int front;

    int rear;

    int size;

    int\* queue;

public:

    CircularQueue(int size) {

        this->front = -1;

        this->rear = -1;

        this->size = size;

        this->queue = new int[size];

    }

    ~CircularQueue() {

        delete[] queue;

    }

    bool isEmpty() {

        return (front == -1 && rear == -1);

    }

    bool isFull() {

        return ((rear+1) % size == front);

    }

    void enqueue(int value) {

        if (isFull()) {

            cout << "Queue is full." << endl;

            return;

        }

        else if (isEmpty()) {

            front = 0;

            rear = 0;

        }

        else {

            rear = (rear+1) % size;

        }

        queue[rear] = value;

    }

    void printPairs() {

        if (isEmpty()) {

            cout << "Queue is empty." << endl;

            return;

        }

        cout << "Pairs of adjacent elements in the queue: ";

        for (int i = front; i != rear; i = (i+1) % size) {

            cout << "(" << queue[i] << ", " << queue[(i+1) % size] << ") ";

        }

        cout << "(" << queue[rear] << ", " << queue[front] << ")" << endl;

    }

};

// main function

int main() {

    CircularQueue queue(5);

    queue.enqueue(10);

    queue.enqueue(20);

    queue.enqueue(30);

    queue.enqueue(40);

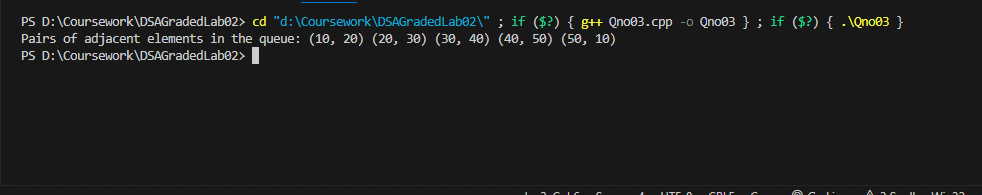
    queue.enqueue(50);

    queue.printPairs();

    return 0;

}

**Output:**



**Task04**

//Taaha Hussain Khan

//L1F21BSCS0917

//D12

#include <iostream>

#include <string>

using namespace std;

class Stack {

private:

    char\* data;

    int top;

    int size;

public:

    Stack(int max\_size) {

        size = max\_size;

        data = new char[size];

        top = -1;

    }

    ~Stack() {

        delete[] data;

    }

    bool isEmpty() {

        return (top == -1);

    }

    bool isFull() {

        return (top == size-1);

    }

    void push(char value) {

        if (isFull()) {

            cout << "Stack overflow." << endl;

            return;

        }

        top++;

        data[top] = value;

    }

    void pop() {

        if (isEmpty()) {

            cout << "Stack underflow." << endl;

            return;

        }

        top--;

    }

    char peek() {

        if (isEmpty()) {

            cout << "Stack is empty." << endl;

            return '\0';

        }

        return data[top];

    }

};

string removeDuplicates(string str) {

    Stack st(str.length());

    for (char ch : str) {

        if (!st.isEmpty() && st.peek() == ch) {

            st.pop();

        }

        else {

            st.push(ch);

        }

    }

    string result = "";

    while (!st.isEmpty()) {

        result = st.peek() + result;

        st.pop();

    }

    return result;

}

int main() {

    string str = "abbzyzt";

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

    str = removeDuplicates(str);

    cout << "String after removing pairs of duplicates: " << str << endl;

    return 0;

}

**Output:**

