LAB TASK # 05

CODE # 01:

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

#define max 1000

class Queue {
private:
    int front, rear;
    int arr[max];

public:
    Queue() {
        front = -1;
        rear = -1;
    }

    bool enqueue(int value) {
        if (rear >= max - 1) {
            cout << "Queue overflow" << endl;
            return false;
        } else {
            if (front == -1) {
                front = 0;
            }
            rear++;
```

```
        arr[rear] = value;
        cout << value << " enqueued into queue." << endl;
        return true;
    }
}

bool dequeue() {
    if (front == -1) {
        cout << "Queue underflow" << endl;
        return false;
    } else {
        cout << "Dequeued element is = " << arr[front] << endl;
        front++;

        if (front > rear) {
            front = rear = -1;
        }
        return true;
    }
}

bool isEmpty() {
    return front == -1;
}

void display() {
    if (isEmpty()) {
        cout << "Queue is empty" << endl;
```

```
        } else {
            cout << "Queue elements are: ";
            for (int i = front; i <= rear; i++) {
                cout << arr[i] << " ";
            }
            cout << endl;
        }
    }
};

int main() {
    Queue q;

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);
    q.enqueue(40);

    q.display();

    q.dequeue();
    q.display();

    if (q.isEmpty()) {
        cout << "Queue is empty." << endl;
    } else {
        cout << "Queue is not empty." << endl;
    }
```

```
    return 0;
}
```

CODE # 02:

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

const int MAX = 1000;

int main() {
    char input[MAX];
    char words[MAX][MAX];
    int wordLengths[MAX];

    // Taking input
    cout << "Enter a string (end input with Enter): ";
    cin.getline(input, MAX);

    int wordCount = 0;
    int index = 0;

    for (int i = 0; input[i] != '\0'; i++) {
        if (input[i] != ' ') {
            words[wordCount][index++] = input[i];
        } else {
            if (index > 0) {
                wordLengths[wordCount] = index;
```

```
            words[wordCount][index] = '\0';
            wordCount++;
            index = 0;
        }
    }
}

if (index > 0) {
    wordLengths[wordCount] = index;
    words[wordCount][index] = '\0';
    wordCount++;
}

char result[MAX];
int resultIndex = 0;

for (int j = 0; j < wordCount; j++) {
    for (int k = 0; k < wordLengths[j]; k++) {
        result[resultIndex++] = words[j][k];
    }
    if (j < wordCount - 1) {
        result[resultIndex++] = ' ';
    }
}

result[resultIndex] = '\0';

cout << "Concatenated Result: " << result << endl;
```

```
    return 0;
}
```