**Assignment 1 - Finals**
**Data Structures and Algorithm**

CODE:

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

#define MAX_SIZE 5

class Stack {
private:
   int arr[MAX_SIZE];
   int top;

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

   void push(int value) {
      if (isFull()) {
         cout << "Stack Overflow!" << endl;
         return;
      }
      arr[++top] = value;
      cout << "Pushed " << value << " into the stack." << endl;
      display();
   }

   void pop() {
      if (isEmpty()) {
         cout << "Stack Underflow!" << endl;
         return;
      }
      cout << "Popped " << arr[top--] << " from the stack." << endl;
      display();
   }

   void peek() {
      if (isEmpty()) {
```

```
            cout << "Stack is empty." << endl;
            return;
        }
        cout << "Peek value: " << arr[top] << endl;
        display();
    }

    void count() {
        cout << "Total number of items in the stack: " << top + 1 << endl;
        display();
    }

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

    bool isFull() {
        return top == MAX_SIZE - 1;
    }

    void clear() {
        while (!isEmpty()) {
            pop();  // Pop all elements to clear the stack
        }
        cout << "Stack cleared." << endl;
    }

    void display() {
        cout << "All values in the stack are:" << endl;
        for (int i = top; i >= 0; i--) {
            cout << "\t| " << arr[i] << " |" << endl;
        }
    }
};

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

public:
    Queue() {
        front = -1;
```

```
    rear = -1;
}

void enqueue(int val) {
    if (isFull()) {
        cout << "Queue is full. Cannot enqueue more items." << endl;
        return;
    }
    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear = (rear + 1) % MAX_SIZE; // Update rear in a circular manner
    }
    arr[rear] = val;
    cout << "Enqueued " << val << endl;
    display();
}

int dequeue() {
    if (isEmpty()) {
        cout << "Queue is empty. Cannot dequeue." << endl;
        return -1;
    }
    int dequeued = arr[front];
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % MAX_SIZE; // Update front in a circular manner
    }
    cout << "Dequeued " << dequeued << endl;
    display();
    return dequeued;
}

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

bool isFull() {
    return (rear + 1) % MAX_SIZE == front;
}

int peek() {
    if (isEmpty()) {
```

```
            cout << "Queue is empty. Cannot peek." << endl;
            return -1;
        }
        return arr[front];
    }

    int count() {
        if (isEmpty()) {
            return 0;
        } else if (front <= rear) {
            return rear - front + 1;
        } else {
            return MAX_SIZE - front + rear + 1;
        }
    }

    void clear() {
        front = rear = -1;
        cout << "Queue is cleared." << endl;
        display();
    }

    void display() {
        cout << "All Values in the Queue are:" << endl;
        cout << "\t\t\t";
        if (isEmpty()) {
            cout << "| | | | | |" << endl;
            return;
        }
        int index = front;
        do {
            cout << "| " << arr[index] << " | ";
            index = (index + 1) % MAX_SIZE;
        } while (index != (rear + 1) % MAX_SIZE);
        cout << endl;
    }
};

int main() {
    int choice;
    string input;
    char *endPtr;

    while (true) {
```

```
cout << "Select data structure you want to use:" << endl;
cout << " [1] Stack" << endl;
cout << " [2] Circular Queue" << endl;
cout << " [0] Exit" << endl;
cout << "Enter your choice: ";
cin >> input;
choice = strtol(input.c_str(), &endPtr, 10);

if (*endPtr != '\0' || cin.fail()) {
    cout << "Invalid input. Please enter a number." << endl;
    cin.clear();
    cin.ignore(numeric_limits
    <streamsize>::max(), '\n');
    continue;
}

switch (choice) {
    case 1: {
        Stack stack;
        int stackChoice, value;
        do {
            cout << "Select operation you want to perform:" << endl;
            cout << " [1] Push" << endl;
            cout << " [2] Pop" << endl;
            cout << " [3] IsEmpty()" << endl;
            cout << " [4] IsFull()" << endl;
            cout << " [5] Peek" << endl;
            cout << " [6] Count" << endl;
            cout << " [7] Clear" << endl;
            cout << " [0] Exit" << endl;
            cin >> stackChoice;

            switch (stackChoice) {
                case 1:
                    if (!stack.isFull()) {
                        cout << "Enter an item to push into the stack: ";
                        cin >> value;
                        stack.push(value);
                    } else {
                        cout << "Stack overflow! Cannot push more items." << endl;
                    }
                    break;
                case 2:
                    if (!stack.isEmpty()) {
```

```
                stack.pop();
            } else {
                cout << "Stack underflow! Cannot pop." << endl;
            }
            break;
        case 3:
            if (stack.isEmpty()) {
                cout << "Stack is empty." << endl;
            } else {
                cout << "Stack is not empty." << endl;
            }
            break;
        case 4:
            if (stack.isFull()) {
                cout << "Stack is full." << endl;
            } else {
                cout << "Stack is not full." << endl;
            }
            break;
        case 5:
            if (!stack.isEmpty()) {
                stack.peek();
            } else {
                cout << "Stack is empty. Cannot peek." << endl;
            }
            break;
        case 6:
            stack.count();
            break;
        case 7:
            stack.clear();
            break;
        case 0:
            cout << "Exiting stack operations." << endl;
            break;
        default:
            cout << "Invalid choice!" << endl;
      }
   } while (stackChoice != 0);
   break;
}
case 2: {
   Queue queue;
   int queueChoice, val;
```

```
do {
    cout << "Select operation you want to perform:" << endl;
    cout << " [1] Enqueue" << endl;
    cout << " [2] Dequeue" << endl;
    cout << " [3] IsEmpty" << endl;
    cout << " [4] IsFull()" << endl;
    cout << " [5] Peek()" << endl;
    cout << " [6] Count()" << endl;
    cout << " [7] Clear" << endl;
    cout << " [0] Exit" << endl;
    cin >> queueChoice;

    switch (queueChoice) {
        case 1: {
            cout << "You Selected Enqueue Operation" << endl;
            cout << "Enter an Item to Enqueue in Queue: ";
            cin >> val;
            if (cin.fail()) {
                cout << "Invalid input. Please enter a valid integer." << endl;
                cin.clear();
                cin.ignore(numeric_limits
                        <streamsize>::max(), '\n');
                    continue;
                } else {
                    queue.enqueue(val);
                    break;
                }
            }
            case 2:
                cout << "You Selected Dequeue Function" << endl;
                queue.dequeue();
                break;
            case 3:
                cout << "You Selected IsEmpty Operation" << endl;
                if (queue.isEmpty())
                    cout << "The Queue is Empty" << endl;
                else
                    cout << "The Queue is not Empty" << endl;
                queue.display();
                break;
            case 4:
                cout << "You Selected IsFull Operation" << endl;
                if (queue.isFull())
                    cout << "The Queue is Full" << endl;
```

```
                    else
                        cout << "The Queue is not Full" << endl;
                    queue.display();
                    break;
                case 5:
                    cout << "Peek Function Called" << endl;
                    cout << "The Peek Value is: " << queue.peek() << endl;
                    queue.display();
                    break;
                case 6:
                    cout << "Count Function Called" << endl;
                    cout << "Total Number of Items in the Queue are: " << queue.count() <<
endl;
                    queue.display();
                    break;
                case 7:
                    queue.clear();
                    break;
                case 0:
                    cout << "Exiting queue operations." << endl;
                    break;
                default:
                    cout << "Invalid choice. Please try again." << endl;
                    break;
                }
            } while (queueChoice != 0);
            break;
        }
        case 0:
            cout << "Exiting program." << endl;
            return 0;
        default:
            cout << "Invalid choice! Please select again." << endl;
    }
  }

  return 0;
}
```