```cpp
#include "q3.h"

//main suroutine


int main()
{
    srand((unsigned)time(0));
    int choice = 0;



    Queue readyQueue;       //creating object for Ready Queue
    Queue jobQueue;         //creating object for Job Queue
    Queue waitQueue;        //creating object for Waiting Queue


    //entry of element for Ready Queue
    cout << "How many elemets for Ready Queue: " << endl;
    cin >> choice;

    while (choice != 0)
    {
        readyQueue.enqueue((rand()%100)+100);
        choice--;
    }


    //entry of element for Job Queue
    cout << "How many elemets for Job Queue: " << endl;
    cin >> choice;

    while (choice != 0)
    {
        jobQueue.enqueue((rand()%100)+100);
        choice--;
    }


    //entry of element for Waiting Queue
    cout << "How many elemets for Waiting Queue: " << endl;
    cin >> choice;

    while (choice != 0)
    {
        waitQueue.enqueue((rand()%100)+100);
        choice--;
    }

    //display Ready Queue
    cout << "\nReady Queue with process IDs (PIDs): ";
    readyQueue.queueDisplay();

    //display Job Queue
    cout << "\nJob Queue with process IDs (PIDs):   ";
    jobQueue.queueDisplay();

    //display Waiting Queue
    cout << "\nWaiting Queue with process IDs (PIDs):   ";
    waitQueue.queueDisplay();

    cout << "\n\n" << endl;


    choice = 0;
    cout << "\tMenu:\n\n" << endl;
        cout << "\t1. Scheduler\n\t2. Interrupt\n\t3. I/O Event Wait\n\t4. I/O Event Completion\n\n" <<
```

```
endl;

        cout << "Enter your choice(1,2,3,4 or -1 for exit): ";
        cin >> choice;
        cout << "\n" << endl;

    int tmpEl = 0;

    while(choice != -1)
    {
        switch(choice)
        {
            case 1: //Scheduler Dispatch
            {
                if(readyQueue.isEmpty())
                {
                    cout << "Ready Queue Empty.\nScheduler Dispatch can't take place!\n";
                    break;
                }
                jobQueue.enqueue(readyQueue.dequeue());


                cout << "All the Queues after Scheduler Dispatch:\n\n" << endl;


                //display Ready Queue
                cout << "\nReady Queue with process IDs (PIDs): ";
                readyQueue.queueDisplay();

                //display Job Queue
                cout << "\nJob Queue with process IDs (PIDs):   ";
                jobQueue.queueDisplay();

                //display Waiting Queue
                cout << "\nWaiting Queue with process IDs (PIDs):   ";
                waitQueue.queueDisplay();

                cout << "\n\n" << endl;
                break;
            }
            case 2: //Interrupt
            {

                if(jobQueue.isEmpty())
                {
                    cout << "Job Queue Empty.\nInterrupt can't take place!\n";
                    break;
                }

                readyQueue.enqueue(jobQueue.dequeue());


                cout << "All the Queues after Interrupt:\n\n" << endl;


                //display Ready Queue
                cout << "\nReady Queue with process IDs (PIDs): ";
                readyQueue.queueDisplay();

                //display Job Queue
                cout << "\nJob Queue with process IDs (PIDs):   ";
                jobQueue.queueDisplay();

                //display Waiting Queue
                cout << "\nWaiting Queue with process IDs (PIDs):   ";
                waitQueue.queueDisplay();
```

```
            cout << "\n\n" << endl;
            break;
        }
        case 3: //I\0 Event Wait
        {

            if(jobQueue.isEmpty())
            {
                cout << "Job Queue Empty.\nEvent Wait can't take place!\n";
                break;
            }

            waitQueue.enqueue(jobQueue.dequeue());


            cout << "All the Queues after I/O Event Wait:\n\n" << endl;


            //display Ready Queue
            cout << "\nReady Queue with process IDs (PIDs): ";
            readyQueue.queueDisplay();

            //display Job Queue
            cout << "\nJob Queue with process IDs (PIDs):   ";
            jobQueue.queueDisplay();

            //display Waiting Queue
            cout << "\nWaiting Queue with process IDs (PIDs):   ";
            waitQueue.queueDisplay();

            cout << "\n\n" << endl;
            break;
        }
        case 4: //I\0 Event Completion
        {

            if(waitQueue.isEmpty())
            {
                cout << "Waiting Queue Empty.\nEvent Completion can't take place!\n";
                break;
            }

            readyQueue.enqueue(waitQueue.dequeue());


            cout << "All the Queues after I/O Event Completion:\n\n" << endl;


            //display Ready Queue
            cout << "\nReady Queue with process IDs (PIDs): ";
            readyQueue.queueDisplay();

            //display Job Queue
            cout << "\nJob Queue with process IDs (PIDs):   ";
            jobQueue.queueDisplay();

            //display Waiting Queue
            cout << "\nWaiting Queue with process IDs (PIDs):   ";
            waitQueue.queueDisplay();

            cout << "\n\n" << endl;
            break;
        }
        default:
            cout << "Invalid Choice! Please try again.\n\n" << endl;
    }
```

```
        cout << "Menu:\n\n" << endl;
        cout << "\t1. Scheduler\n\t2. Interrupt\n\t3. I/O Event Wait\n\t4. I/O Event Completion\n\n" <<
endl;

        cout << "Enter your choice(1,2,3,4 or -1 for exit): ";
        cin >> choice;
        cout << "\n" << endl;

    }


    cout << "All the Queues after Scheduler Dispatch, Interrupt and I/O Event\n\n" << endl;

    //display Ready Queue
    cout << "\nReady Queue with process IDs (PIDs): ";
    readyQueue.queueDisplay();

    //display Job Queue
    cout << "\nJob Queue with process IDs (PIDs):   ";
    jobQueue.queueDisplay();

    //display Waiting Queue
    cout << "\nWaiting Queue with process IDs (PIDs):   ";
    waitQueue.queueDisplay();

    cout << "\n\n" << endl;


    return 0;
}
```