COSC 220 - Project 2

Fall 2021

Team: John Meyers, Nick Krisulevicz

Contents: Mini Library and Driver Programs

Code contains comments and designations for author for respective programs

Files:

miniDList.h
miniDList.cpp
miniListStack.h
miniListStack.cpp
miniListQueue.h
miniListQueue.cpp
miniArrStack.h
miniArrStack.cpp
miniArrQueue.h
miniArrQueue.cpp
student.h
student.cpp
driver.cpp
makefile

Readme:

Welcome to the Mini Library! What you will find in these classes are methods to implement the stack, queue, and list data structures normally found within the C++ Standard Template Library. The programs included are miniature versions of their STL counterparts, but are functional using any kind of data type with the template instantiation they are designed with! The design of these mini classes is intended to plug in any primitive or class data type and allow a stack, queue, or list of these data types to be created and used at your convenience.

The problem specifically solved with this instance of the Mini Library was to manage numerous student objects using the Student Class from Project 1. This is demonstrated using a driver executable that allows the user to insert, remove and print elements from the stacks, queues and lists. The library also comes with a makefile which allows quick compilation of the source code into executable format. Simply type “make” without the quotes into your terminal and your executable is ready to go! Then, you will be able to manage the students as you please.

That is just a mere example of the countless things you can do with your Mini Library!

In the case of the student application, the inputs will be directly from the user from a numbered menu of actions to perform. Entering the personal information of the student and deleting them will affect the outputs presented on screen. The references to each location in the stacks and queues depends on the application. The Mini Library has the capability to implement your data structure in either Array or Linked List format! The linked lists contain dynamically allocated memory which will be freed once a node is deleted, but the array uses static memory which will remain in use until the program is finished. The small, lightweight programs should run lightning quick and should not use up excessive memory when they are run. All in all, the Mini Library should be a useful tool available when trying to perform any application involving stacks, queues, or lists.

This Mini Library was created by John Meyers and Nick Krisulevicz, and none of the four bonus opportunities were attempted.

Source Code:

miniDList.h:

```
#ifndef MINIDLIST_h
#define MINIDLIST_h
#include <ostream>
template <class DataType>
class miniDList{
private:
struct DNode{ // doubly linked list node
       DataType data; //  data stored in node
       DNode* next = nullptr; //  next node in list
       DNode* prev = nullptr; //  previous node  in list

};

 DNode* trailer = nullptr;
 DNode* header = nullptr; // list sentinels
public: miniDList(){                    // constructor
       header = new DNode;
       //trailer = nullptr;
       //header = nullptr; // list sentinels
       trailer  = new DNode;         //have them point to each other
       header->next = trailer;
       header->prev = nullptr;
       trailer->prev = header;
       trailer->next = nullptr;
}

miniDList(const   miniDList& l); // Copy constructor
~miniDList(); // Destructor
const DataType& front() const;      // get front element
const DataType& back() const;       // get back element
void   removeFront(); // Remove & return the front
void   removeBack();             // Remove & return the back
void addFront(const DataType& e);       // Add to the front
void addBack(const DataType& e);           // Add to the back
int   size()   const; // Returns the number of elements in list
 bool contains(const DataType& e);   // Tests for membership
```

```
void  display()  const; // Prints the elements of list
 miniDList<DataType>& operator=(const miniDList<DataType>& l);
friend std::ostream& operator<<(std::ostream &out, const
miniDList<DataType>& d){
      DNode* curr = d.header->next;
      out << "List: " << std::endl;
      while(curr){
        out << curr->data << std::endl;
        curr = curr->next;
      }
      out << std::endl;
      return out;
    }
 //Overloaded assignment
//std::ostream& operator<<(std::ostream& os, const DNode& dnode);
protected:                          //local utilities
void add(DNode * v, const DataType& e); //insert new node  before  v
                           void remove(DNode* v);     //remove node v
};
//#include "miniDList.cpp"
#endif
```

miniDList.cpp:

```
//written by John Meyers
//COSC-220
//this is the linked list functions to make the linked list
//more so to implement the linked lists into the queue and stack
#include <iostream>
using namespace std;
#include "miniDList.h"
template <class DataType>
miniDList<DataType>::miniDList(const miniDList& l){
    DNode * curr = l.header->next;
    int count = 0;
    while(curr != trailer){
        curr = curr->next;
        count++;
    }
```

```
    curr = l.header->next;
    DNode * temp = l.header->next;
    header->next = nullptr;
    trailer->prev = nullptr;
    for(curr = l.header->next; curr != l.trailer; curr = curr->next){
        for(int i = 0; i < count - 1; i++){
            temp = temp->next;
        }
        addFront(temp->data);
        temp = l.header->next;
        count--;
    }
}
template <class DataType>
miniDList<DataType>::~miniDList(){
    DNode * curr = nullptr;
    DNode * temp = nullptr;
    curr = header;
    temp = curr->next;
    while(curr != nullptr){
        delete curr;
        curr = temp;
        temp = curr->next;
    }

}
template <class DataType>
const DataType& miniDList<DataType>::front() const{

    return header->next->data;
}
template <class DataType>
const DataType& miniDList<DataType>::back() const{

    return trailer->prev->data;
}
template <class DataType>
void miniDList<DataType>::removeFront(){
    remove(header->next);
}
```

```
template <class DataType>
void miniDList<DataType>::removeBack(){
    remove(trailer->prev);
}
template <class DataType>
void miniDList<DataType>::addFront(const DataType& e){
    add(header->next, e);
}
template <class DataType>
void miniDList<DataType>::addBack(const DataType& e){

    add(trailer, e);
}
template <class DataType>
int miniDList<DataType>::size() const{
    int count = 0;
    DNode * curr = nullptr;
    if(header->next == trailer){
        return 0;
    }
    else if(header->next  == nullptr){
        return 1;
    }
    curr = header->next;
    while(curr != trailer){
        curr = curr->next;
        count++;
    }
    return count;

}
template <class DataType>
bool miniDList<DataType>::contains(const DataType& e){
    DNode * curr = header->next;
    if(e == curr->data){
        return true;
    }
    while(curr->next != nullptr){
        curr =  curr->next;
    }
```

```
    if(e == curr->data)
        return true;
    else
        return false;
}
template <class DataType>
void miniDList<DataType>::display() const{
    int count =  1;
    if(header->next != trailer){
    DNode * curr = header->next;
    while(curr != trailer){
        cout << "Node " << count  << ":" << curr->data << endl;
        count++;
        curr = curr->next;
    }
    }
    else{
        cout << "there is nothing to print" << endl;
    }



}
template <class DataType>
miniDList<DataType>& miniDList<DataType>::operator=(const
miniDList<DataType>& l){
    int count = 0;
    if(this == &l){
        DNode* temp = nullptr;
        return *this;
    }else{
        DNode* curr = this->header->next;
        while(curr){
            DNode* nex = curr->next;
            delete curr;
            curr = nex;
        }
        curr = l.header->next;
        while(curr){
            count++;
```

```
            curr = curr->next;
        }
        curr = l.header->next;
        DNode* temp = l.header->next;
        this->header->next = nullptr;
        this->trailer->prev = nullptr;

        for(curr = l.header->next; curr != l.trailer; curr = curr->next){
            for(int i = 0; i < count - 1; i++){
                temp = temp->next;
            }
            this->addFront(temp->data);
            temp = l.header->next;
            count--;
        }
        return *this;
    }


}
template <class DataType>
void miniDList<DataType>::add(DNode * v, const DataType& e){
    DNode * curr = nullptr;
    DNode * newNode = new DNode;
    newNode->data = e;
    newNode->next = nullptr;
    newNode->prev = nullptr;
    if(header->next == trailer){

        header->next = newNode;
        newNode->prev = header;
        newNode->next = trailer;
        trailer->prev = newNode;
        return;
    }


    if(header->next == v){

        curr = header->next;
```

```
        header->next = newNode;
        newNode->next = curr;
        curr->prev = newNode;
        return;
    }

    else if(trailer == v){
        curr = trailer->prev;
        trailer->prev = newNode;
        newNode->prev = curr;
        curr->next = newNode;
        newNode->next = trailer;

    else{
        return;//this should not happen
    }

}
template <class DataType>
void miniDList<DataType>::remove(DNode* v){
    DNode * curr = nullptr;
    if(header->next == trailer){
        return;
    }
    if(header->next == v && trailer->prev == v){
        if(header->next == v){
        header->next = trailer;
        trailer->prev = header;
        delete v;
        }
        return;
    }
    else if(trailer->prev == v){
        curr = v->prev;
        if(trailer->prev == v){
            trailer->prev = curr;
            curr->next  = trailer;
            delete v;
        }
    }
```

```
    else if(header->next == v){
        curr = v->next;
        if(header->next == v){
            header->next = curr;
            curr->prev = header;
            delete v;
        }
    }

    else {
        cout << "uh oh" << endl;
        return;
    }
```

miniListStack.h:

```
#ifndef MINILISTSTACK_H
#define MINILISTSTACK_H
#include "miniDList.h"
template <class DataType>
 class miniListStack{
private:
const int returnInt = -999999999;
miniDList<DataType> sList;
        /**
For  the  list-backed  implementation:
*
*
For the array-based:
*
enum {DEF_CAPACITY =100};  //default stack capacity
DataType* arr; // The array of items
int capacity; // The size of the current array
int top; // The location of the top element
*/ public:

        miniListStack();        // Constructor for ListStack
        miniListStack(const miniListStack & l);  // Copy contstructor
        ~miniListStack(); // Destructor
        int  size()  const; // get the number of elements in the stack
```

```
        bool isEmpty() const; // Check if the stack is empty
        const DataType& top() const;       //get the top ememento without
popping it
        void push(const DataType& e); // Pushes  an  object  onto  the
stack
        void pop(); // Pop an object off the stack
        void  printStack()  const; // Prints the stack from the top, down

miniListStack<DataType>& operator=(const  miniListStack<DataType>& l);  //
Assignment  operator
};
//#include "miniListStack.cpp"             //
#endif
```

miniListStack.cpp

```
//written by John Meyers
//COSC-220
//this is for the linked lists and implementing it into the stack class
#include <iostream>
#include "miniListStack.h"
using namespace std;


template <class DataType>
miniListStack<DataType>::miniListStack(){

}       // Constructor for ListStack
template <class DataType>
miniListStack<DataType>::miniListStack(const miniListStack & l){
      *this = l;
}
// Copy contstructor
template <class DataType>
miniListStack<DataType>::~miniListStack(){
}
// Destructor
template <class DataType>
int  miniListStack<DataType>::size()  const{
```

```
        return sList.size();
}       // get the number of elements in the stack
template <class DataType>
bool miniListStack<DataType>::isEmpty() const{
        if(sList.front() == -1){
                return true;
        }
        else
                return false;
}       // Check if the stack is empty
template <class DataType>
const DataType& miniListStack<DataType>::top() const{
        if(isEmpty() == false){
        return sList.front();
        }
        return sList.front();
        //return returnInt;
}       //get the top emement without popping it
template <class DataType>
void miniListStack<DataType>::push(const DataType& e){
       sList.addFront(e);
       return;
}       // Pushes  an  object  onto  the  stack
template <class DataType>
void miniListStack<DataType>::pop(){
      sList.removeFront();
}       // Pop an object off the stack
template <class DataType>
void  miniListStack<DataType>::printStack()  const{
        sList.display();
        return;
}
template <class DataType>
miniListStack<DataType>& miniListStack<DataType>::operator=(const
miniListStack<DataType>& l){
        int count = 0;
        miniListStack<DataType>& temp;
        if(this == &l){
                return *this;
        }else{
```

```
                while(this.top()){
                        this.pop();
                }

                while(l.top()){
                        temp.push(l.top());
                        l.pop();
                        count++;
                }
                while(temp.top()){
                        this.push(temp.top());
                        temp.pop();
                }
                /*curr = l.header;
                DNode* temp = l.header;
                this->header = nullptr;
                this->trailer = nullptr;

                for(curr = l.header; curr; curr = curr->next){
                        for(int i = 0; i < count - 1; i++){
                                temp = temp->next;
                        }
                        this->addFront(temp->data);
                        temp = l.header;
                        count--;
                }*/
                return *this;
        }

}
```

miniListQueue.h:

```
#ifndef MINILISTQUEUE_H
#define MINILISTQUEUE_H
#include "miniDList.h"

template <class DataType>
```

```
 class miniListQueue {
    private:
    miniDList<DataType> list;
/**
For  the  list-backed  implementation:
*
*
For the array-based:
*
DataType* arr;  // The array of items
int capacity; // The size of the current array
int front; // The location of the front element
int rear; // The location of the rear element
*/
 public:
miniListQueue ();  // Constructor
miniListQueue(const miniListQueue & l); // Copy Constructor
~miniListQueue(); //   Destructor
int   size()   const; // get the number of elements in the queue
 bool isEmpty() const; // Check if the queue is empty
void  enqueue(const  DataType& e);  // Enqueue element at rear
void  dequeue(); // dequeue element at front
const DataType& front() const; //return the front element but not remove
void  printQueue()  const; // Prints the queue from the front to the rear
 miniListQueue<DataType>& operator=(const miniListQueue<DataType>& l); //
Assignment operator
};
//#include "miniListQueue.cpp"
#endif
```

miniListQueue.cpp:

```
//written by john meyers
//COSC-220
//this is the cpp for the linked lists and implementing them into the
queue
#include <iostream>
#include "miniListQueue.h"
using namespace std;
```

```
template <class DataType>
miniListQueue<DataType>::miniListQueue(){

}   // Constructor
template <class DataType>
miniListQueue<DataType>::miniListQueue(const miniListQueue & l){
       *this = l;
}   // Copy Constructor
template <class DataType>
miniListQueue<DataType>::~miniListQueue(){

}   //   Destructor
template <class DataType>
int  miniListQueue<DataType>::size()   const{
       return list.size();
}   // get the number of elements in the queue
template <class DataType>
bool miniListQueue<DataType>::isEmpty() const{

    if(list.front() == -1){
           return false;
       }
       else{
           return true;
       }

}   // Check if the queue is empty
template <class DataType>
void  miniListQueue<DataType>::enqueue(const  DataType& e){
     list.addBack(e);
}   // Enqueue element at rear
template <class DataType>
void  miniListQueue<DataType>::dequeue() {
       list.removeFront();
}   // dequeue element at front
template <class DataType>
const DataType& miniListQueue<DataType>::front() const{
       return list.front();
```

```
}   //return the front element but not remove
template <class DataType>
void  miniListQueue<DataType>::printQueue()  const{
       /*if(isEmpty() == false){
           cout << "nothing is in the queue" << endl;
           return;
       }
       else{*/
    list.display();
    return;
    //}
}   // Prints the queue from the front to the rear
template <class DataType>
 miniListQueue<DataType>& miniListQueue<DataType>::operator=(const
miniListQueue<DataType>& l){
    if(this == &l){
                return *this;
        }else{
        while(this.front()){
            this.dequeue();
        }
        while(l.front()){
            this.enqueue(l.front());
            l.dequeue();
        }
        return *this;
    }
 }
```

miniArrStack.h:

```
//Nick Krisulevicz
//Teammate: John Meyers
//COSC 220 - Project 2
//11/12/2021
//miniArrStack.h

#ifndef MINIARRSTACK_H
#define MINIARRSTACK_H
```

```
//#include "miniDList.h"
#include <iostream>
using namespace std;

enum {DEF_CAPACITY =100};  //default stack capacity
template <class DataType>
class miniArrStack{

private:
    //For  the  list-backed  implementation:
//  miniDList<DataType> list;

    //For the array-based:
    //enum {DEF_CAPACITY =100};  //default stack capacity

    DataType* arr; // The array of items

    int capacity; // The size of the current array

    int top; // The location of the top element

    void enlarge(); //Private function to expand the boundaries of the
array

public:
        miniArrStack (int cap); // Constructor for ArrStack

    miniArrStack(); // Constructor for ListStack

    miniArrStack(const miniArrStack &);  // Copy contstructor

    ~miniArrStack(); // Destructor

    int  size()  const; // get the number of elements in the stack

        bool isEmpty() const; // Check if the stack is empty

        const DataType& printTop(); // const throw(StackEmpty); //get the
top emement without popping it
```

```
        void push(const DataType& e); // throw(StackFull); // Pushes  an
object  onto  the  stack

        void pop(); // throw(StackEmpty); // Pop an object off the stack

        void printStack()  const; // Prints the stack from the top, down

        miniArrStack<DataType>& operator=(const  miniArrStack<DataType>&
rhs);  //  Assignment  operator

    friend std::ostream& operator<<(std::ostream &out, const
miniArrStack<DataType>& d){
        std::cout << "Stack: " << std::endl;
        for(int i = 0; i < d.capacity; i++){
            out << d.arr[i] << std::endl;
        }
            out << std::endl;
            return out;
    }
};
//#include "miniStack.cpp" //for template instantiation
#endif
```

miniArrStack.cpp:

```
//Nick Krisulevicz
//Teammate: John Meyers
//COSC 220 - Project 2
//11/12/2021
//miniArrStack.cpp

#include "miniArrStack.h"
#include <iostream>
using namespace std;

template<class DataType>
miniArrStack<DataType>::miniArrStack(int cap){ //constructor which
initializes a stack given a capacity passed in
    capacity = cap;
```

```
    arr = new DataType[capacity];
    top = capacity - 1;
}

template<class DataType>
miniArrStack<DataType>::miniArrStack(){ //default constructor
    arr = new DataType[100];
    capacity = 100;
    top = 0;
}

template<class DataType>
miniArrStack<DataType>::miniArrStack(const miniArrStack & e){ //copy
contructor
    arr = new DataType[e.capacity];
    capacity = e.capacity;
    top = -1;
    for(int i = 0; i < e.top; i++){
        push(e.arr[i]);
    }
}

template<class DataType>
miniArrStack<DataType>::~miniArrStack(){ //destructor
    delete [] arr;
}

template <class DataType>
int miniArrStack<DataType>::size() const{ //function to get the size of
the stack
    int index = 0;
    index = top + 1;
    return index;
}

template <class DataType>
bool miniArrStack<DataType>::isEmpty() const{ //function to determine
whether or not the stack is empty
    bool status;
    if (top == -1)
```

```
        status = true;
    else
        status = false;
    return status;
}

template <class DataType>
const DataType& miniArrStack<DataType>::printTop(){ //function to print
the top element of the stack, without deleting it
    if(isEmpty() == true){
        cout << "Stack is empty" << endl;
        return arr[top];
    }
    else{
        cout << arr[top];
        return arr[top];
    }
}

template <class DataType>
void miniArrStack<DataType>::push(const DataType& e){ //function to insert
an item on the top of the stack
    if(top == capacity){
        cout << "Stack is full" << endl;
    }
    else{
        top++;
        arr[top] = e;
    }
}

template <class DataType>
void miniArrStack<DataType>::pop(){ //function to remove the top item from
the stack
    if(isEmpty()){
        cout << "Stack is empty, nothing to pop" << endl;
    }
    else{
        top--;
    }
```

```
}

template <class DataType>
void miniArrStack<DataType>::printStack() const{ //function to print all
items in the stack in order
    int index = top;
    if(isEmpty() == true){
        cout << "Stack is empty, nothing to print" << endl;
    }
    else{
        while(index != 0){
            cout << arr[index] << endl;
            index --;
        }
    }
}

template <class DataType>
miniArrStack<DataType>& miniArrStack<DataType>::operator= (const
miniArrStack<DataType>& rhs){ //operator overload to allow assignment of
class objects
    delete[] this->arr;
    capacity = rhs.capacity;
    top = rhs.top;
    this->arr = new DataType[capacity];
    for(int i = 0; i < capacity; i++){
        this->arr[i] = rhs.arr[i];
    }
    return *this;
}

template<class DataType>
void miniArrStack<DataType>::enlarge(){ //private function to allow the
capacity of the stack array to increase if needed
    capacity *= 2;
    DataType* chestpain = new DataType[capacity];
    for (int i = 0; i < capacity; i++){
        chestpain[i] = arr[i];
    }
    delete [] arr;
```

```
    arr = chestpain;
}
```

miniArrQueue.h:

```
//Nick Krisulevicz
//Teammate: John Meyers
//COSC 220 - Project 2
//11/12/2021
//miniArrQueue.h

#ifndef MINIARRQUEUE_H
#define MINIARRQUEUE_H
#include "miniDList.h"
#include <iostream>
using namespace std;

template <class DataType>
class miniArrQueue {

private:
    //For  the  list-backed  implementation:
    miniDList<DataType> list;

    //For the array-based:
    DataType* arr;  // The array of items

    int capacity; // The size of the current array

    int front; // The location of the front element

    int rear; // The location of the rear element

    int count; //Current size of the queue

public:
    miniArrQueue ();  // Constructor

    miniArrQueue(const miniArrQueue & e); // Copy Constructor
```

```
    ~miniArrQueue(); //   Destructor

    int size()   const; // get the number of elements in the queue

    bool isEmpty() const; // Check if the queue is empty

    void  enqueue(const  DataType& e);  // Enqueue element at rear

    void  dequeue(); // throw(QueueEmpty); // dequeue element at front

    const DataType& printFront(); //const; //throw(QueueEmpty); //return
the front element but not remove

    void  printQueue(); //  const; // Prints the queue from the front to
the rear

    miniArrQueue<DataType>& operator=(const miniArrQueue<DataType>& rhs);
// Assignment operator

    friend std::ostream& operator<<(std::ostream &out, const
miniArrQueue<DataType>& d){ //operator<< overload to allow use with class
objects
        if(d.front == d.rear){
            out << "Queue is Empty" << std::endl;
            return out;
        }else{
            out << "Queue: ";
            for(int i = 0; i < d.rear; i++){
                out << d.arr[i] << " ";
            }
            out << std::endl;
            return out;
        }
    }
};
//#include "miniArrQueue.cpp"              //for template instantiation
#endif
```

miniArrQueue.cpp:

```cpp
//Nick Krisulevicz
//Teammate: John Meyers
//COSC 220 - Project 2
//11/12/2021
//miniArrQueue.cpp

#include "miniArrQueue.h"
#include <iostream>

using namespace std;

template<class DataType>
miniArrQueue<DataType>::miniArrQueue(){ //constructor that initializes
queue with default values
    capacity = 100;
    arr = new DataType[capacity];
    front = 0;
    rear = 1;
}

template<class DataType>
miniArrQueue<DataType>::miniArrQueue(const miniArrQueue & e){ //copy
constructor which initializes a new queue with the original values in a
new reference location
    arr = new DataType[e.capacity];
    capacity = e.capacity;
    front = e.front;
    rear = e.rear;
    for (int i = 0; i < e.rear; i++){
        enqueue(e.arr[i]); //deep copy
    }
}

template<class DataType>
miniArrQueue<DataType>::~miniArrQueue(){ //destructor
    delete [] arr;
}
```

```
template <class DataType>
int miniArrQueue<DataType>::size() const{
    return rear + 1; //function to return the size of the queue
}

template <class DataType>
bool miniArrQueue<DataType>::isEmpty() const{ //function used to return
whether or not the queue is empty
    bool status;
    if(front == -1){
        status = true;
    }
    else{
        status = false;
    }
    return status;
}

template <class DataType>
void miniArrQueue<DataType>::enqueue(const DataType& e){ //function to
enqueue an item, inserts a new item at the rear of the queue
    if(rear > capacity){
        std::cout << "Full" << std::endl;
        return;
    }else{
        arr[rear] = e;
        rear++;
    }
    return;
}

template <class DataType>
void miniArrQueue<DataType>::dequeue(){ //function to dequeue an item from
the queue, removes an item from the front of the queue
    if (isEmpty() == true){
        cout << "Queue is empty, nothing to dequeue" << endl;
    }
    else{
        front = (front + 1) % capacity;
        front++;
```

```
    }
}

template <class DataType>
const DataType& miniArrQueue<DataType>::printFront(){ //function to print
the front element in the queue, without deleting it
    if (isEmpty() == true){
        cout << "Queue is Empty" << endl;
        return arr[front++];
    }
    else{
        cout << arr[front];
        return arr[front++];
    }
}

template <class DataType>
void miniArrQueue<DataType>::printQueue(){ //function to print all the
items in the queue in their order
    if(front == rear){
        std::cout << "Queue is Empty" << std::endl;
        return;
    }else{
        std::cout << "Queue: ";
        for(int i = 0; i < rear; i++){
            std::cout << arr[i] << " ";
        }
        std::cout << std::endl;
        return;
    }
}


template <class DataType>
miniArrQueue<DataType>& miniArrQueue<DataType>::operator=(const
miniArrQueue<DataType>& rhs){ //operator overload function to allow
assignment of queue objects
    if(this == &rhs){
        return *this;
    }
```

```
    delete[] arr;
    arr = new DataType[rhs.capacity];
    capacity = rhs.capacity;
    front = -1;
    rear = -1;
    for(int i = 0; i < rhs.rear; i++){
        enqueue(rhs.arr[i]);
    }
    return *this;
}
```

student.h:

```
//JOhn Meyers
//Cosc-220
//the function declarations for the student objects
#ifndef STUDENT_H
#define STUDENT_H
#include <iostream>
#include <ostream>
#include "book.h"
using namespace std;
class student{
    public:
        string year;
        string name;
        int idNumber;
        book * list;
        void removebook(string booktitle);
        book * createBook();
        student * next;
        student * prev;
        student();
        ~student();
        string getname() const;
        int getidNumber() const;
        void printbooklist();
        bool operator==(const student& s) const;
        bool operator==(int temp) const;
```

```
        friend std::ostream& operator<<(std::ostream& os, const student&);

};
#endif
```

student.cpp:

```
//john Meyers
//COSC-220
//this implements the book funtions in order to create removes and other
functions like print
#include "student.h"
#include "book.h"
#include <iostream>
using namespace std;
student::student(){
    list = nullptr;

}
student::~student(){
    book * newbook;
    book * curr;
    newbook = list;
    while(newbook != nullptr){
        curr = newbook->nextbook;
        delete newbook;
        newbook = curr;
    }
}

book * student::createBook(){
    book * newBook = new book();
    book * bookptr;
    string t, d, r;
    cout << "what is the title of the book?" << endl;
    cin >> newBook->Title;
    cout << "what is the due date of the book?" << endl;
    cin.ignore();
    cin >> newBook->duedate;
```

```
    cout << "is the book renewable?" << endl;
    cin.ignore();
    cin >> newBook->renewable;
    if(list == nullptr){
            list = newBook; //if there is no head, make the first book
inserted point to head
        }
        else{
            bookptr = list; //if there are books in the list, point the
new book to the last book
            while(bookptr->nextbook){
                    bookptr = bookptr->nextbook; //while loop to proceed
through the list until it finds the end
        }
        bookptr->nextbook = newBook;
    }
    return newBook;
}
void student::removebook(string booktitle){



    book * prevbook;
    book * bookptr;

    if(list == nullptr){ //if list is empty, nothing will be deleted
    return;
    }
    cout << list->Title << endl;
    if(list->Title == booktitle){ //if the first book is the target one,
it will be deleted

    bookptr = list->nextbook;
    delete list;
    list = bookptr;
    return;
    }
    else{ //iterate through the list until the book is found and delete it
        bookptr = list;
```

```
    while(bookptr != nullptr && bookptr->Title != booktitle){ //while loop
to proceed through list
        prevbook = bookptr;
        bookptr = bookptr->nextbook;

    }
    if(bookptr){
        prevbook->nextbook = bookptr->nextbook;
        delete bookptr;

        }else{
        cout<<"Book not found"<<endl;
        }
    }
}




void student::printbooklist(){
    book *bookptr;
    bookptr = list;
    while(bookptr){
        cout << bookptr->Title << ", ";
        cout << bookptr->duedate << ", ";
        cout << bookptr->renewable << endl;
        bookptr = bookptr->nextbook;
    }//prints out the book list for the specific student
}
bool student::operator==(const student& s) const{
    if(idNumber == s.idNumber){
        return true;
    }
    else{
        return false;
    }

}
bool student::operator==(int temp) const{
    if(this->idNumber == temp){
```

```
        return true;
    }
    else{
        return false;
    }
}

std::ostream& operator<<(std::ostream& os, const student& s){
    os << s.getname() << " id number:" <<  s.getidNumber() << endl;
    return os;
}
string student::getname() const{
    return name;
}
int student::getidNumber() const{
    return idNumber;
}
```

driver.cpp:

```
// written by John Meyers
//COSC-220
#include "miniDList.cpp"
#include "miniListStack.cpp"
#include "miniListQueue.cpp"
#include "miniArrStack.cpp"
#include "miniArrQueue.cpp"
#include "studentDlist.h"
#include <iostream>
#include <stack>
using namespace std;



int main(){
    char answer;
    int count = 0;
    int count2 = 0;
    miniListStack<student> stack;
```

```
    miniListQueue<student> queue;
    miniArrStack<student> ArrStack;
    miniArrQueue<student> ArrQueue;
    cout << "Would you like to enter the program?(Y/N) " << endl;
    cin >> answer;
    if(answer == 'N'){
                cout << " have a good day!" << endl;
        }
        else if(answer == 'Y'){
        int menu = 0;
        student s, s2, s3, s4;
        while(menu != -1){
            menu = 0;
            cout << "-1. to exit the program" << endl;
            cout << "list functions:" << endl;
            cout << "1. push an element onto the list stack" << endl;
            cout << "2. pop an element off of the list stack" << endl;
            cout << "3. prints the top of the list stack" << endl;
            cout << "4. prints the list stack" << endl;
            cout << "5. enqueue an element onto the list queue" << endl;
            cout << "6. dequeue an element off of the list queue" << endl;
            cout << "7. prints the first element in the list queue" <<
endl;
            cout << "8. prints the list queue" << endl;
            cout << "array functions:" << endl;
            cout << "9. push an element onto the array stack" << endl;
                        cout << "10. pop an element off of the array
stack" << endl;
                        cout << "11. prints the top of the array stack" <<
endl;
                        cout << "12. prints the array stack" << endl;
                        cout << "13. enqueue an element onto the array
queue" << endl;
                        cout << "14. dequeue an element off of the array
queue" << endl;
                        cout << "15. prints the first element in the array
queue" << endl;
                        cout << "16. prints the array queue" << endl;
```

```
            cout << "enter the number in front of the action you want
performed: " << endl;
            cin >> menu;
            if(menu == -1){
                cout << "goodbye!" << endl;
                break;
            }
            else if(menu == 1){
                cout << "what is the students year? " << endl;
                    cin >> s.year;
                    cout << "what is the students name?(no spaces)" <<
endl;
                    cin.ignore();
                cin >>  s.name;
                cout << "what is the students id number? " << endl;
                cin.ignore();
                    cin >> s.idNumber;
                count = 0;
                count++;
                cout << "now pushing the student onto the stack" << endl;
                stack.push(s);
            }
            else if(menu == 2){
                count--;
                stack.pop();
            }
            else if(menu == 3) {
                if(count >= 1){

                student temp = stack.top();
                cout << "the top of the current list stack is: " <<
temp.name << ", id number: " << temp.idNumber << endl;
                }
                else{
                    cout << "the list stack is empty" << endl;
                }

            }
            else if(menu == 4){
                stack.printStack();
```

```
            }
            else if(menu == 5){
                cout << "what is the students year? " << endl;
                                cin >> s2.year;
                                cout << "what is the students name?(no
spaces)" << endl;
                                cin.ignore();
                                cin >>  s2.name;
                                cout << "what is the students id number? "
<< endl;
                                cin.ignore();
                                cin >> s2.idNumber;
                count2 = 0;
                count2++;
                cout << "now pushing the student onto the queue" << endl;
                queue.enqueue(s2);
            }
            else if(menu == 6){
                queue.dequeue();
                count2--;
            }
            else if(menu == 7){
                if(count2 >= 1){
                student temp = queue.front();
                cout << "the first element of the current list queue is: "
<< temp.name << ", id number: " << temp.idNumber << endl;

                }
                else{
                    cout << "the list queue is empty" << endl;
                }
            }
            else if(menu == 8){
                queue.printQueue();
            }
            else if(menu == 9){
                cout << "what is the students year? " << endl;
                                cin >> s3.year;
                                cout << "what is the students name?(no
spaces)" << endl;
```

```
                                cin.ignore();
                                cin >>  s3.name;
                                cout << "what is the students id number? "
<< endl;
                                cin.ignore();
                                cin >> s3.idNumber;
                                cout << "now pushing the student onto the
stack" << endl;
                ArrStack.push(s3);
            }
            else if(menu == 10){
                ArrStack.pop();
            }
            else if(menu == 11){
                ArrStack.printTop();
            }
            else if(menu == 12){
                ArrStack.printStack();
            }
            else if(menu == 13){
                cout << "what is the students year? " << endl;
                                cin >> s4.year;
                                cout << "what is the students name?(no
spaces)" << endl;
                                cin.ignore();
                                cin >>  s4.name;
                                cout << "what is the students id number? "
<< endl;
                                cin.ignore();
                                cin >> s4.idNumber;
                cout << "now pushing the student onto the queue" << endl;
                ArrQueue.enqueue(s4);
            }
            else if(menu == 14){
                ArrQueue.dequeue();
            }
            else if(menu == 15){
                ArrQueue.printFront();
            }
            else if(menu == 16){
```

```
                ArrQueue.printQueue();
            }
            else{
                cout << "Invalid input" << endl;
            }
        }
    }
    else{
        cout << "invalid input goodbye!" << endl;
    }
}
```

makefile:

```
all: miniDriver
miniDriver: miniDriver.o miniListStack.o miniListQueue.o miniDList.o
student.o book.o studentDlist.o miniArrStack.o miniArrQueue.o
    g++ -o miniDriver miniDriver.o miniListStack.o miniListQueue.o
miniDList.o student.o book.o studentDlist.o miniArrStack.o miniArrQueue.o
miniDriver.o: miniDriver.cpp
    g++ -c miniDriver.cpp
miniListStack.o: miniListStack.cpp
    g++ -c miniListStack.cpp
miniListQueue.o: miniListQueue.cpp
    g++ -c miniListQueue.cpp
miniDList.o: miniDList.cpp
    g++ -c miniDList.cpp
student.o: student.cpp
    g++ -c student.cpp
book.o: book.cpp
    g++ -c book.cpp
studentDlist.o: studentDlist.cpp
    g++ -c studentDlist.cpp
miniArrStack.o: miniArrStack.cpp
    g++ -c miniArrStack.cpp
miniArrQueue.o: miniArrQueue.cpp
    g++ -c miniArrQueue.cpp
clean:
    rm *o miniDriver
```

Outputs:

Test #1:

```
Would you like to enter the program?(Y/N)
Y
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
1
what is the students year?
rs
what is the students name?(no spaces)
john
what is the students id number?
30
now pushing the student onto the stack
-1. to exit the program
list functions:
```

1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
1
what is the students year?
sophmore
what is the students name?(no spaces)
nick
what is the students id number?
1
now pushing the student onto the stack
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack

12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
1
what is the students year?
junior
what is the students name?(no spaces)
mo
what is the students id number?
50
now pushing the student onto the stack
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
4
Node 1:mo id number:50

Node 2:nick id number:1

Node 3:john id number:30

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
3
the top of the current list stack is: mo, id number: 50
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue

15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
2
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
4
Node 1:nick id number:1

Node 2:john id number:30

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:

9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
2
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
3
the top of the current list stack is: john, id number: 30
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue

7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
3
the top of the current list stack is: john, id number: 30
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
2
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack

4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
4
there is nothing to print
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
3
the list stack is empty
-1. to exit the program

list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
2
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:

-1
Goodbye!
Test #2:
Would you like to enter the program?(Y/N)
Y
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
2
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack

12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
6
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
there is nothing to print
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:

9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
4
there is nothing to print
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
-1
goodbye!

Test #3:
Would you like to enter the program?(Y/N)
Y
-1. to exit the program
list functions:
1. push an element onto the list stack

2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
5
what is the students year?
sophmore
what is the students name?(no spaces)
john
what is the students id number?
2
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack

13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:john id number:2

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
5
what is the students year?
freshman
what is the students name?(no spaces)
mark
what is the students id number?
30
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack

3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
5
what is the students year?
freshman
what is the students name?(no spaces)
keanan
what is the students id number?
45
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue

14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:john id number:2

Node 2:mark id number:30

Node 3:keanan id number:45

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
6
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue

7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:mark id number:30

Node 2:keanan id number:45

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
6
-1. to exit the program
list functions:

1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:keanan id number:45

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue

enter the number in front of the action you want performed:
5
what is the students year?
junior
what is the students name?(no spaces)
hannah
what is the students id number?
93
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:keanan id number:45

Node 2:hannah id number:93

Test #4:
Would you like to enter the program?(Y/N)
Y
-1. to exit the program
list functions:

1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
5
what is the students year?
sophmore
what is the students name?(no spaces)
john
what is the students id number?
20
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack

12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
5
what is the students year?
junior
what is the students name?(no spaces)
rebecca
what is the students id number?
3092
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
7
the first element of the current list queue is: john, id number: 20
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack

3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:john id number:20

Node 2:rebecca id number:3092

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue

enter the number in front of the action you want performed:
6
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
8
Node 1:rebecca id number:3092

Test #5:
Would you like to enter the program?(Y/N)
Y
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack

10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
9
what is the students year?
sophmore
what is the students name?(no spaces)
nate
what is the students id number?
20
now pushing the student onto the stack
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
9
what is the students year?
junior
what is the students name?(no spaces)

john
what is the students id number?
60
now pushing the student onto the stack
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
9
what is the students year?
freshman
what is the students name?(no spaces)
mar
what is the students id number?
256
now pushing the student onto the stack
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue

7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
12
mar id number:256

john id number:60

nate id number:20

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:

12
mar id number:256

john id number:60

nate id number:20

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
11
mar id number:256
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:

9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
10
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
11
john id number:60
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue

7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
12
john id number:60

nate id number:20

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
10
-1. to exit the program
list functions:

1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
12
nate id number:20

-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue

enter the number in front of the action you want performed:
11
nate id number:20
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
10
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue

14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
12
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
-1
goodbye!

Test #6:
Would you like to enter the program?(Y/N)
Y
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue

8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
13
what is the students year?
sophmore
what is the students name?(no spaces)
seth
what is the students id number?
29
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
13

what is the students year?
senior
what is the students name?(no spaces)
sean
what is the students id number?
78
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
13
what is the students year?
middle
what is the students name?(no spaces)
nate
what is the students id number?
29
now pushing the student onto the queue
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack

4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
15
 id number:0
-1. to exit the program
list functions:
1. push an element onto the list stack
2. pop an element off of the list stack
3. prints the top of the list stack
4. prints the list stack
5. enqueue an element onto the list queue
6. dequeue an element off of the list queue
7. prints the first element in the list queue
8. prints the list queue
array functions:
9. push an element onto the array stack
10. pop an element off of the array stack
11. prints the top of the array stack
12. prints the array stack
13. enqueue an element onto the array queue
14. dequeue an element off of the array queue
15. prints the first element in the array queue
16. prints the array queue
enter the number in front of the action you want performed:
16
Queue: id number:0
 seth id number:29

sean id number:78
nate id number:29