1. Consider the problems of 1) rendering a collection of playing cards on a computer screen and 2) allowing individual cards to be selected with mouse clicks. Regard a computer screen to be a matrix of pixels, each of which has a location that is specified by horizontal and vertical coordinates with respect to the upper left corner of the screen (the origin). Each card occupies a rectangular space on the screen, with central axis perpendicular to the vertical. The width (horizontal extent) of a card, in pixels, is given by parameter \( \text{wid} \), and the height (vertical extent), in pixels, is given by parameter \( \text{hgt} \). The location of a card is determined by the coordinates \((\text{left}, \text{top})\) of the leftmost, uppermost pixel the card occupies.

The following describes rules for the rendering and selection of cards. Consider the rendering policy first. Cards may overlap (two or more cards may occupy the same pixel). Only the topmost card in an overlapping region is visible. To achieve this, the rendering of all cards is accomplished by drawing the bottommost card first, then the next to bottommost card and so on. Thus, when the rendering is completed, an observer will see portions of the topmost card of each overlapping region occluding all other cards overlapping that region. Now consider the selection policy. When the mouse is clicked, the coordinates of the mouse cursor are used to determine which, if any, card is selected. If one or more cards occupy the clicked pixel, the topmost card occupying that pixel is the one selected. Any selected card becomes the topmost of all cards.

The class below is used to create cards:

```java
class Card {
    int left; // current pixels in from left (horizontal coordinate)
    int top; // current pixels down from top (vertical coordinate)
    int wid; // width of card in pixels
    int hgt; // height of card in pixels
    int suit_rank; // suit and rank of card #/4=rank #/4 = suit

    public:
    Card (int suit_rank, int left, int top) {
        this->suit_rank = suit_rank;
        this->left = left;
        this->top = top;
        wid = 60;
        hgt = 120;
    }

    // returns true iff mouse is over card (x and y are mouse coordinates)
    bool mouseOver (int x, int y) {
        if (x > left && left+wid > x && y > top && top+hgt > y) return true;
        else return false;
    }

    void renderCard() { // Rendering code for card appears here }
};
```

To facilitate rendering and selection we use the following two partially specified classes:
class dNode {
public:
    Card *card;
    dNode *next, *prev;
    dNode(Card *card) { this->card = card; this->next = this->prev = NULL; }
};

class Dequeue {
    dNode *head, *tail;

public:
    Dequeue() { head = tail = NULL; }
    Dequeue &copy();
    Dequeue &select(int,int);
    void addAtFrontOfDequeue(Card*) {
        if (head == NULL)
            head = tail = new dNode(card);
        else {
            dNode *temp = new dNode(card);
            temp->next = head;
            head->prev = temp;
            head = temp;
        }
    }
    void renderAll();
};

Observe that the class Dequeue implements a doubly linked list of dNode objects, each of which references a Card object. The copy() method is intended to make and return a deep copy of the list. Making a deep copy entails making a copy of the dNode list structure, creating variables head and tail but assigning those variables addresses corresponding to dNodes in the new list, and duplicating Card addresses in the new dNodes. The select(int,int) method takes as input horizontal and vertical mouse coordinates, determines a selected card using the over method of the Card class, and adjusts the list structure to comply with the selection policy stated earlier. It outputs the results Dequeue object. The renderAll() method renders the cards according to the rendering policy stated above using the renderCard() method of the Card class to render each card.

a. [10 points] Implement copy().

b. [10 points] Implement select(int,int).

c. [10 points] Implement renderAll().