CS 101 Lab 3 **Summer 2016** **Name ________________________**

A deck is a linear data structure that allows inserting and removing items at both ends.

First we consider implementing a deck using a circular array representation.
Complete the seven methods of class ArrayDeck so that each method runs in O(1) time.

```
class ArrayDeck {
        int *array;
        int front, back, capacity, n;
public:
        ArrayDeck (int cap);
        bool isEmpty( );
        bool isFull( );
        void insertFront (int x);
        void insertBack (int x);
        int removeFront( );
        int removeBack( );
};
```

```
ArrayDeck::ArrayDeck (int cap) {
        array=new int[cap];

}
```

```
bool ArrayDeck::isEmpty( ) {

}
```

```
bool ArrayDeck::isFull( ) {

}
```

```
void ArrayDeck::insertFront (int x) {
        if (isFull( )) return;

}
```

```
void ArrayDeck::insertBack (int x) {
        if (isFull( )) return;

}
```

```
int ArrayDeck::removeFront( ) {
        if (isEmpty( )) return 0;

}
```

```
int ArrayDeck::removeBack( ) {
        if (isEmpty( )) return 0;

}
```

Next we consider implementing a deck using a doubly-linked list representation.
Here our doubly-linked list is non-circular and it does not have any header node.
Complete the six methods of class LinkedDeck so that each method runs in O(1) time.

```
class LinkedDeck {
	struct Node {
		int data;
		Node *prev, *next;
	};
	Node *front, *back;
public:
	LinkedDeck( );
	bool isEmpty( );
	void insertFront (int x);
	void insertBack (int x);
	int removeFront( );
	int removeBack( );
};
```

```
LinkedDeck::LinkedDeck( ) {



}
```

```
bool LinkedDeck::isEmpty( ) {
	return front==NULL && back==NULL;
}
```

```
void LinkedDeck::insertFront (int x) {
	Node *temp=new Node;



}
```

```
void LinkedDeck::insertBack (int x) {
	Node *temp=new Node;



}
```

```
int LinkedDeck::removeFront( ) {
	if (isEmpty( )) return 0;
	int x=front->data;
	Node *temp=front;



	delete temp;
	return x;
}
```

```
int LinkedDeck::removeBack( ) {
	if (isEmpty( )) return 0;
	int x=back->data;
	Node *temp=back;



	delete temp;
	return x;
}
```