ELEC 278
Fundamentals of Information Structures

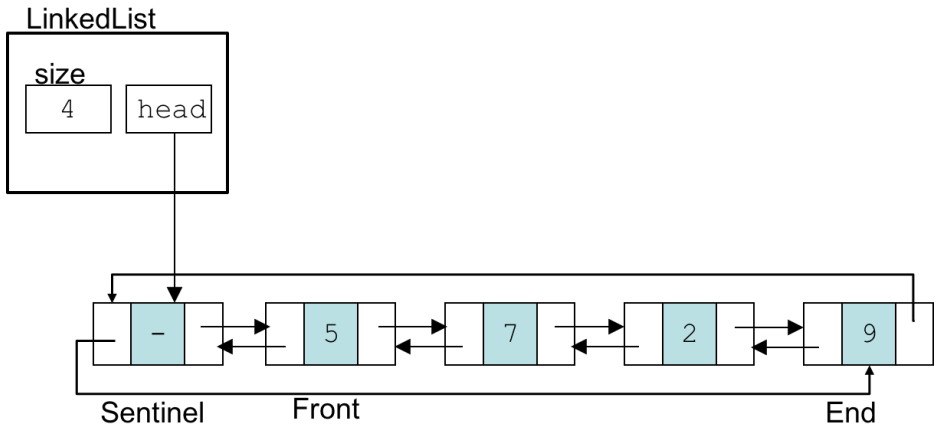
Fall 2016
Section 001

Assignment 2

Suggested completion date:
Oct 11th 2016

Problem 1

Consider the following Linked Lists design:



The structure has the following properties which should hold after any operation on the list:

1. The ***head*** always points to the **Sentinel**
2. **Sentinel** is a dummy node that has no value.
3. Sentinel's ***next*** pointer points to the front of the list.
4. Sentinel's ***prev*** pointer points to the end of the list.
5. The ***prev*** of the first node in the list points to the **Sentinel**
6. The ***next*** of the last node in the list points to the **Sentinel**
7. **Sentinel** points to itself if the list is empty

The definition of the linked list is as follows:

```
typedef struct Node{
    double value;
    struct Node* next;
    struct Node* prev;
}Node;

typedef struct LinkedList{
    int size;
    Node* head;
} LinkedList;
```

Write the code for the following functions:

1. Construct and initialize
```
void init(LinkedList* l){…}
```

2. Insert new value to the front
```
void addFront(LinkedList* l, int x) {…}
```

3. Insert new value to the end
```
void addEnd(LinkedList* l, int x) {…}
```

4. Remove the front of the list
```
void removeFront(LinkedList* l) {…}
```

5. Remove the end of the list
```
void removeEnd(LinkedList* l) {…}
```

6. Print the list
```
void print(LinkedList* l) {…}
```

Problem 2

Consider the code written in problem 1. Write a complete **Stack** structure that uses only the above functions to do the following:

1. Construct and initialize
```
void initS(Stack* s){…}
```

2. Push
```
void push(Stack* s, int x) {…}
```

3. Pop
```
int pop(Stack* s, int* res) {…}
```

4. Check if the stack is empty. Returns 1 if empty and 0 if not.
```
int isEmptyS(Stack* s) {…}
```

Problem 3

Consider the code written in problem 1. Write a complete **Queue** structure that uses only the above functions to do the following:

1. Construct and initialize
```
void initQ(Queue* q){…}
```

2. Enqueue
```
void enqueue(Queue* q, int x) {…}
```

3. Dequeue
```
int dequeue(Queue* q, int* res) {…}
```

4. Check if the queue is empty. Returns 1 if empty and 0 if not.
```
int isEmptyS(Queue* q) {…}
```

Problem 4

Consider the code written in problem 1. Write a complete **Deque** structure that uses only the above functions to do the following:

1. Construct and initialize
```
void initDQ(Deque* d){…}
```

2. EnqueueHead
```
void enqueueHead(Deque* d, int x) {…}
```

3. DequeueHead
```
int dequeueHead(Deque* d, int* res) {…}
```

4. EnqueueTail
```
void enqueueTail(Deque* d, int x) {…}
```

5. DequeueTail
```
int dequeueTail(Deque* d, int* res) {…}
```

Problem 5

Consider the code written in problems 1,2 and 3 and the following *main* function.

```
int main(){

    Stack s;
    Queue q;
    initS(&s);
    initQ(&q);
    int res = -1;
    pop(&s,&res);
    int i=0;
    for (i=0; i<10; i+=2) {
        push(&s,i);
    }
    pop(&s,&res);
    while(!isEmptyS(&s)){
        pop(&s,&res);
        enqueue(&q,res);
    }
    while(!isEmptyQ(&q)){
        dequeue(&q,&res);
        push(&s,res);
    }
    return 0;
```

What is the content of the stack **s** after executing the above code? What does this code do?

Problem 6

Consider the implementation of Deque over arrays. Write the code for functions:

1. EnqueueHead
2. DequeueTail

Problem 7

Using only Stacks and Queues structures and the operations mentioned in Problem 2 and 3. Write a function that takes a pointer to stack **A** and returns two pointers to stacks **odd** and **even**. The first stack should contain all the odd numbers of stack A in reverse order. The second stack should contain all the even numbers in order.

Example: