CS137
Data Structure

Homework on March 16, 2020

Prof : Jialiang Lu

TA : Lei Wang

Part I – ADT of List

Question n° 1 : Create a set of file DLListMain.cpp, DLList.h and List.h to define your **Doubly Linked List**. Initially, only the List.h file contained the following code:

```
template <class Elem> class List {
public:
  virtual void clear() = 0;
  virtual bool insert(const Elem&) = 0;
  virtual bool append(const Elem&) = 0;
  virtual bool remove(Elem&) = 0;
  virtual void setStart() = 0;
  virtual void setEnd() = 0;
  virtual void prev() = 0;
  virtual void next() = 0;
  virtual int leftLength() const = 0;
  virtual int rightLength() const = 0;
  virtual bool setPos(int pos) = 0;
  virtual bool getValue(Elem&) const = 0;
  virtual void print() const = 0;
};
```

Explain the function of following part of this **ADT** :

template / virtual / clear() = 0 / insert(const Elem&) / leftLength() const = 0

Question n°2 : Create a class of Doubly linked list node （named **Link**） in DLList.h.

```
template <class Elem> class Link {









};
```

Partie II – Implementation of doubly linked list

***Question n°*3** : Complete the private part of this class DLList in DLList.h.

```
template <class Elem> class DLList: public List<Elem> {
private:
        Link<Elem>* head; // Point to list header
        Link<Elem>* tail; // Pointer to last Elem in list
        Link<Elem>* fence;// Last element on left side
        int leftcnt;      // Size of left partition
        int rightcnt;     // Size of right partition
        void init() {     // Initialization routine
        //TO BE COMPLETED
        }
        void removeall() {   // Return link nodes to free store
        //TO BE COMPLETED
        }
public:
        DLList(int size=DEFAULT_LIST_SIZE);
        ~DLList(); // Destructor
        void clear();
        bool insert(const Elem&);
        bool append(const Elem&);
        bool remove(Elem&);
        void setStart();
        void setEnd();
        void prev();
        void next();
        int leftLength() const;
        int rightLength() const;
        bool setPos(int pos);
        bool getValue(Elem& it) const;
        void print () const;
};
```

Question n°4 : Complete the implementation of each function in public.

Partie III – Main function

***Question n°* 5** : Give some examples and test your code in DLListMain.cpp, explain reasons of your input choice.

Partie IV– Freelist

Question n°6 : Create a class of Doubly linked list node in DLList.h as following:

```
template <class Elem> class Link {
private:
    static Link<Elem>* freelist; // Head of the freelist; for all Link objects
public:
    Elem element; // Value for this node
    Link *next;   // Pointer to next node
    Link *prev;   // Pointer to previous node
    Link(const Elem& e, Link* prevp=NULL, Link* nextp=NULL)
    Link(Link* prevp=NULL, Link* nextp=NULL)
    // Overload new & delete operators for freelist
    void* operator new(size_t);
    void operator delete(void*);
};
```

Question n°7 (Bonus) : Complete the Initialization of freelist, Function overloading and Operator overloading.

```
template <class Elem>
Link<Elem>* Link<Elem>::freelist
template <class Elem>
Link<Elem>:: Link(const Elem& e, Link* prevp=NULL, Link* nextp=NULL)
template <class Elem>
Link<Elem>:: Link(Link* prevp=NULL, Link* nextp=NULL)
template <class Elem>
void* Link<Elem>::operator new(size_t)
template <class Elem>
void Link<Elem>::operator delete(void* ptr)
```

Question n°8 (Bonus) : Explain **the use of （void* ptr）** in **delete**.

Question n°9 (Bonus) : Explain **the use of freelist (Advantage)**.