Solutions des exercices de cours: Pointeurs, variables dynamiques et listes chaînées

1. Écrire un sous-programme qui permet de supprimer tous les nœuds d'une liste chaînée.

```
// delete all the nodes from the list.
void destroyList(ptr &head){
	while(head){
		ptr p = head;
		head = head->next;
		delete p;
	}
}
```

2. Écrire un sous-programme qui détermine si une liste chaînée est vide. Le sous-programme doit retourner true si la liste est vide, sinon il retourne false.

```
// determine whether the list is empty.
// returns true if the list is empty, otherwise returns false.
bool isEmpty(ptr head){
     return (head == NULL);
}
```

3. Écrire un sous-programme qui détermine le nombre de nœuds dans une liste chaînée.

```
// length of a linked list.
// returns the number of nodes in the list.
int length(ptr head){
	int count = 0;
	while(head){
		count++;
		head = head->next;
	}
	return count;
}
```

4. Écrire une fonction qui permet de déterminer si un élément figure dans une liste chaînée triée. La fonction retourne un pointeur sur l'élément, si l'élément existe dans la liste, sinon elle retourne le pointeur NULL.

```
// search an item in a sorted linked list.
ptr search(ptr head, int data){
	while(head && head->data < data) { head = head->next; }

	if(head && head->data == data){ return head; }
	else { return NULL; }
}
```

5. Écrire un sous-programme qui élimine les éléments redondants dans une liste simplement chaînée d'entiers.

```
void removeDuplicates(ptr head){
        while (head != NULL){
            ptr prev = head;
            ptr q = head->next;
            while (q != NULL){
                if (q->data != head->data){
                     prev = q;
                     q = q->next;
                }
                else {
                    prev->next = q->next;
                    delete q;
                    q = prev->next;
                }
            }
            head = head->next;
         }
}
```

6. Écrire un sous-programme qui permet d'ajouter un élément (nombre) à une liste simplement chaînée triée d'entiers. Si le nombre existe déjà le sous-programme ne doit rien faire.

```
void addItem(ptr& head, int item){
       ptr prev = NULL;
       ptr q = head;

       while (q != NULL && q->data < item){
              prev = q;
              q = q->next;
       }

       if (q == NULL or q->data > item){
              ptr p = new node;
              p->data = item;
              p->next = q;

              if (prev != NULL){ prev->next = p; }
              else { head = p;}
       }
}
```