Started on	Monday, 18 September 2023, 4:52 PM
State	Finished
Completed on	Thursday, 21 September 2023, 12:49 PM
Time taken	2 days 19 hours
Marks	11.00/11.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

Mark 1.00 out of 1.00

Assume that class AList uses an array to implement the list. The private fields of this class are declared as below:

```
template <typename T>
class AList :public List<T> {
   private:
      const static int MAX = 20; //the maximum members of the list
      T* data; // keep the list
      int cursor = 0; // keep the position of the cursor
      int size; // the real size of the list
  public:
      AList() { data = new T[MAX];}
...
}
```

Write method **void insert(const T& v)** that inserts the new element whose value is v into the position of cursor and moves forward all elements after the cursor. Note that the method does not change the cursor.

Your code starts from line 62.

Answer: (penalty regime: 0 %)

```
1  void insert(const T& v){
2      if ( this->size == MAX){
3          return;
4      }
5      for ( int i = size ; i > cursor ; i--){
6          data[i] = data[i-1];
7      }
8      data[cursor] = v;
9      this->size ++;
10     return;
11
12 }
```

	Test	Expected	Got	
✔	x.printAll();	{2,3,1,4}	{2,3,1,4}	✔
✔	x.printAll();	{3,1,4,2}	{3,1,4,2}	✔
✔	x.printAll();	{3,2,1,4}	{3,2,1,4}	✔
✔	x.printAll();	{3,1,2,4}	{3,1,2,4}	✔
✔	x.printAll();	{2}	{2}	✔
✔	x.printAll();	{7,5,2}	{7,5,2}	✔
✔	x.printAll();	{2,5,7}	{2,5,7}	✔
✔	x.printAll();	{5,3,7,1,4,2}	{5,3,7,1,4,2}	✔
✔	x.printAll();	{3,1,9,4,5,2}	{3,1,9,4,5,2}	✔
✔	x.printAll();	{7,3,1,2,4,9}	{7,3,1,2,4,9}	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 2

Correct

Mark 1.00 out of 1.00

Assume that class AList uses an array to implement the list. The private fields of this class are declared as below:

```
template <typename T>
class AList {
   private:
      const static int MAXSIZE = 20; //the maximum members of the list
      T* data; // keep the list
      int cursor = 0; // keep the position of the cursor
      int listSize; // the real size of the list
  public:
...
}
```

Write method **void insertBefore(const T& v, const T& k)** that inserts the new element v before the first element in the list whose value is k. If there is no value k in the list, value v is NOT inserted into the list and the cursor is unchanged. If the value k is in the list but the current size of the list is MAX, the exception out_of_range must be thrown. If the value v is inserted successfully, the cursor becomes before the new element. For example, let L be <3,|1,4> and L.insertBefore(2,3) makes L become <|2,3,1,4>.

Your code starts from line 63.

Answer: (penalty regime: 0 %)

```
1  void insertBefore(const T& v, const T& k) {
2      if (this->listSize == this->MAXSIZE) {
3          throw out_of_range("") ;
4      }
5      int findK = 0;
6      while (findK < this->listSize && data[findK] != k) {
7          findK++;
8      }
9      if (findK == this->listSize) {
10         return;
11     }
12     for (int i = this->listSize; i > findK; i--) {
13         data[i] = data[i - 1];
14     }
15     data[findK] = v;
16     this->listSize++;
17     cursor = findK;
18 }
```

	Test	Expected	Got	
✔	`x.printAll();`	`{2,3,1,4}`	`{2,3,1,4}`	✔
✔	`x.printAll();`	`{3,2,1,4}`	`{3,2,1,4}`	✔
✔	`x.printAll();`	`{3,1,2,4}`	`{3,1,2,4}`	✔
✔	`x.printAll();`	`{3,1,2,4,5,4}`	`{3,1,2,4,5,4}`	✔
✔	`x.printAll();`	`{3,2,5,2,4,5,4}`	`{3,2,5,2,4,5,4}`	✔
✔	`x.printAll();`	`{3,1,3,2,4,5,4}`	`{3,1,3,2,4,5,4}`	✔
✔	`x.printAll();`	`{1,3,1,5,4,5,4}`	`{1,3,1,5,4,5,4}`	✔
✔	`x.printAll();`	`{3,1,4,5,4}`	`{3,1,4,5,4}`	✔

✔	`try {`
	`x.moveToPos(2);`
	`x.insertBefore(2,7);`
	`x.printAll();`
	`} catch (out_of_range) {`
	`cout << "Out of range";`
	`}`
		`Out of range`	`Out of range`	✔

✔	`try {`
	`x.insertBefore(2,7);`
	`} catch (out_of_range) {`
	`cout << "Out of range 0";`
	`}`
	`try {`
	`x.insertBefore(2,7);`
	`x.printAll();`
	`} catch (out_of_range) {`
	`cout << "Out of range 1";`
	`}`
		`Out of range 1`	`Out of range 1`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 3

Correct

Mark 1.00 out of 1.00

Assume that class AList uses an array to implement the list. The private fields of this class are declared as below:

```
template <typename T>
class AList : public List<T>{
   private:
      const static int MAXSIZE = 20; //the maximum members of the list
     T* data; // keep the list
      int cursor = 0; // keep the position of the cursor
      int listSize; // the real size of the list
  public:
...
}
```

Write method **void remove()** that removes the element at the cursor position and moves backward all elements after the cursor. If the cursor position is after the last element, please throw exception out_of_range("Cannot remove element"). Note that the method does not change the cursor.

Your code starts from line 67.

Answer: (penalty regime: 0 %)

```
void remove(){
    if (this-> cursor >= this-> listSize ){
        throw out_of_range("Cannot remove element");
    }
    for ( int i = this->cursor; i < listSize -1; i++){
        data[i] = data[i + 1];
    }
    this->listSize--;
}
```

	Test	Expected	Got	
✔	`x.printAll();`	`{1,4}`	`{1,4}`	✔
✔	`x.printAll();`	`{3,1}`	`{3,1}`	✔
✔	`x.printAll();`	`{3,4}`	`{3,4}`	✔
✔	`x.printAll();`	`{3,1}`	`{3,1}`	✔
✔	`x.printAll();`	`{4}`	`{4}`	✔
✔	`x.printAll();`	`{}`	`{}`	✔

✔	`try {`
	`x.remove();`
	`} catch (out_of_range& e) {`
	`cout << e.what();`
	`}`
		`Cannot remove element`	`Cannot remove element`	✔

✔	`x.printAll();`	`{3}`	`{3}`	✔
✔	`x.printAll();`	`{1,9}`	`{1,9}`	✔
✔	`x.printAll();`	`{3,2,9}`	`{3,2,9}`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 4

Correct

Mark 1.00 out of 1.00

Based on the declaration of LList given in page 107, write method **void insertBefore(const E& v, const E& k)** that inserts the new element v before the first element in the list whose value is k. If there is no value k in the list, value v is NOT inserted into the list and the cursor is unchanged. If the value v is inserted successfully, the cursor becomes before the new element. For example, let L be <3,|1,4> and L.insertBefore(2,3) makes L become <|2,3,1,4>

Your code starts from line 141.

Answer: (penalty regime: 0 %)

```
1
2  void insertBefore(const E& v, const E& k){
3      Link<E>* tmp = this->head;
4      while ( tmp->next && tmp->next->element != k){
5          tmp = tmp -> next;
6      }
7      if (!(tmp->next)) return;
8
9      tmp->next = new Link<E>(v, tmp->next);
10     curr = tmp;
11
12 }
```

	Test	Expected	Got	
✔	L->print();	< | 2 3 1 4 >	< | 2 3 1 4 >	✔
✔	L->print();	< 3 | 4 5 7 >	< 3 | 4 5 7 >	✔
✔	L->print();	< 3 5 | 6 7 >	< 3 5 | 6 7 >	✔
✔	L->print();	< 3 5 | 7 >	< 3 5 | 7 >	✔

	Test	Expected	Got	
✔	`L->print();`	`< 3 5 | 7 7 >`	`< 3 5 | 7 7 >`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 5

Correct

Mark 1.00 out of 1.00

Based on the implementation of the list given in Figure 4.8, write method **template <typename E> void LList<E>::print() const** that prints all elements of the list together with the cursor. For example, if the list has 3 elements 3, 5 and 7 and the cursor is at the beginning, the printed result is < | 3 5 7 >.

Your code starts from line 144.

Answer: (penalty regime: 0 %)

```
template <typename E>
void LList<E>::print() const{
    Link<E>* tmp = head->next;
    cout<<"< ";
    while(tmp){
        if ( tmp == curr->next){
            cout<<"| "<<tmp->element<<" ";
        }else{
            cout<<tmp->element<<" ";
        }
        tmp = tmp->next;
    }
    if (tmp == curr->next){
        cout<<"| ";
    }
    cout<<">";
}
```

	Test	Expected	Got	
✔	L->print();	< | 3 5 7 >	< | 3 5 7 >	✔
✔	L->print();	< 3 5 7 | >	< 3 5 7 | >	✔
✔	L->print();	< 3 5 | 7 >	< 3 5 | 7 >	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 6

Correct

Mark 1.00 out of 1.00

Given the class LList<E> implemented as Figure 4.8, write method **template <typename E> void LList<E>::reverse()** that reverses the list. After reversing the list, the position of the cursor is unchanged. For example, < | 3 5 7> after reversing, it becomes < | 7 5 3>, or <3 | 5 7> => < 7 | 5 3>, or <3 5 7 |> => <7 5 3 |>.

Your code starts from line 145.

Answer: (penalty regime: 0 %)

```
#include <stack>
template <typename E>
void LList<E>::reverse(){
    stack<E> revStack;
    Link<E>* tmp = head->next;
    while ( tmp){
        E value = tmp->element;
        revStack.push(value);
        tmp = tmp->next;
    }
     Link<E>* tmp2 = head->next;
    while (tmp2) {
        tmp2->element = revStack.top();
        revStack.pop();
        tmp2 = tmp2->next;
    }


}
```

	Test	Expected	Got	
✔	L->reverse(); L->print();	< | 7 5 3 >	< | 7 5 3 >	✔
✔	L->reverse(); L->print();	< 7 5 3 | >	< 7 5 3 | >	✔
✔	L->reverse(); L->print();	< 7 5 | 3 >	< 7 5 | 3 >	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 7

Correct

Mark 1.00 out of 1.00

Based on class List defined on page 98, write function **template <typename E> int count(List<E> &L,E K)** that returns number of elements in K whose value is K. Your code starts from line **79**.

Answer: (penalty regime: 0 %)

```
template<typename E>
int count(List<E> &L, E K){
    int res = 0;
    for ( L.moveToStart(); L.currPos()< L.length(); L.next()){

        if ( L.getValue() == K){
            res++;
        }
    }
    return res;
}
```

	Test	Expected	Got	
✔	`cout << count(x,10);`	2	2	✔
✔	`cout << count(x,22);`	3	3	✔
✔	`cout << count(x,3);`	1	1	✔
✔	`cout << count(x,4);`	0	0	✔
✔	`cout << count(x,94);`	1	1	✔
✔	`cout << count(x,14);`	1	1	✔

	Test	Expected	Got	
✔	`cout << count(x,23);`	`0`	`0`	✔
✔	`cout << count(x,9);`	`1`	`1`	✔
✔	`cout << count(x,100);`	`0`	`0`	✔
✔	`cout << count(x,19);`	`1`	`1`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 8

Correct

Mark 1.00 out of 1.00

Based on the List ADT defined on page 98, write function **template <typename T> void del(List<T> &L,T k)** that deletes all elements in L whose value is k? Your code starts from line 74.

Answer: (penalty regime: 0 %)

```
template <typename T>
void del(List<T> &L, T k){

    L.moveToStart();
    int i = 0;
    while( i < L.length() ){
        if( L.getValue() == k){
            L.remove();
        }else{
            L.next();
            i ++;
        }
    }

}
```

	Test	Expected	Got	
✔	del(x,11);x.printAll();	{3,3,10,94,14,22,22,9,22,10,10}	{3,3,10,94,14,22,22,9,22,10,10}	✔
✔	del(x,10);x.printAll();	{3,3,94,14,22,22,9,22}	{3,3,94,14,22,22,9,22}	✔
✔	del(x,22);x.printAll();	{3,3,94,14,9}	{3,3,94,14,9}	✔
✔	del(x,3);x.printAll();	{94,14,9}	{94,14,9}	✔
✔	del(x,9);x.printAll();	{94,14}	{94,14}	✔
✔	del(x,94);x.printAll();	{14}	{14}	✔

	Test	Expected	Got	
✔	`del(x,15);x.printAll();`	{14}	{14}	✔
✔	`del(x,19);x.printAll();`	{14}	{14}	✔
✔	`del(x,14);x.printAll();`	{}	{}	✔
✔	`del(x,10);x.printAll();`	{}	{}	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 9

Correct

Mark 1.00 out of 1.00

Write function **template <typename T> void DuplicationKiller(List<T> &L)** that removes all elements which appear more than once in the list L. For example, let L be <2,4,2,9,4,2,4>, after DuplicationKiller(L), it becomes <9>

Answer: (penalty regime: 0 %)

```
#include <unordered_map>
template <typename T>
void DuplicationKiller( List<T> &L){
    unordered_map<T, int> HashTabel;
    L.moveToStart();
    for ( int i = 0; i < L.length(); i++){
        HashTabel[L.getValue()] ++;
        L.next();
    }
    L.moveToStart();
    int position = 0;
    while ( position < L.length ()){
        if ( HashTabel[L.getValue()] > 1){
            L.remove();
        }else {
            position ++;
            L.next();
        }
    }

}
```

	Test	Expected	Got	
✔	DuplicationKiller(x);x.printAll();	{3}	{3}	✔
✔	DuplicationKiller(x);x.printAll();	{}	{}	✔
✔	DuplicationKiller(x);x.printAll();	{2}	{2}	✔
✔	DuplicationKiller(x);x.printAll();	{2,9,10,12,5,6}	{2,9,10,12,5,6}	✔
✔	DuplicationKiller(x);x.printAll();	{}	{}	✔
✔	DuplicationKiller(x);x.printAll();	{}	{}	✔

	Test	Expected	Got	
✔	`DuplicationKiller(x);x.printAll();`	`{}`	`{}`	✔
✔	`DuplicationKiller(x);x.printAll();`	`{}`	`{}`	✔
✔	`DuplicationKiller(x);x.printAll();`	`{3,5}`	`{3,5}`	✔
✔	`DuplicationKiller(x);x.printAll();`	`{}`	`{}`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 10

Correct

Mark 1.00 out of 1.00

Based on the List ADT given in Figure 4.1, write a function to merge two lists. The input lists have their elements in sorted order, from lowest to highest. The output list should also be sorted from lowest to highest. Your algorithm should run in linear time on the length of the output list. The return list must be a new list and it should be an object of LList<E> (linked list). The prototype of the function is:

template <typename E>

List<E>* merge(List<E>* in1, List<E>* in2);

Your code starts from line 196.

Answer: (penalty regime: 0 %)

```
1  template <typename E>
2  List<E>* merge( List<E>* in1, List<E> * in2){
3      List<E>* res = new LList<E>();
4
5      in1->moveToStart(); in2->moveToStart();
6      while ( in1->currPos() < in1->length() && in2->currPos() < in2->length()){
7          if ( in1->getValue() <= in2->getValue()){
8
9                  res->append( in1->getValue());
10                 in1->next();
11
12         }
13         else{
14
15                 res->append(in2->getValue());
16                 in2->next();
17
18         }
19     }
20     while( in1->currPos() < in1->length()){
21         res->append(in1->getValue());
22         in1->next();
```

	Test	Expected	Got	
✔	l3->print();	< | 2 3 4 6 7 9 >	< | 2 3 4 6 7 9 >	✔
✔	l3->print();	< | 3 7 9 >	< | 3 7 9 >	✔
✔	l3->print();	< | 2 4 6 >	< | 2 4 6 >	✔

	Test	Expected	Got	
✔	`l3->print();`	`< | 3 7 9 12 14 16 >`	`< | 3 7 9 12 14 16 >`	✔
✔	`l3 -> print();`	`< | 2 4 6 13 17 19 >`	`< | 2 4 6 13 17 19 >`	✔
✔	`l3->print()`	`< | 2 4 6 13 17 19 >`	`< | 2 4 6 13 17 19 >`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question 11

Correct

Mark 1.00 out of 1.00

As in the previous question, when a task is added into expression classes, new methods are added into these classes. Please change the way these classes are implemented in such a way that these classes do not change their contents when new tasks are added into these classes:

- Define class Eval to calculate the value of an expression

- Define class PrintPrefix to return the string corresponding to the expression in prefix format

- Define class PrintPostfix to return the string corresponding to the expression in postfix format

Let x be a pointer to an object expressing an expression and eval, pre, pos be the pointers to an object of Eval(), PrintPrefix() PrintPostfix(), respectively, x->accept(eval) or eval->visit(x) will return the value of the expression x, x->accept(pre) or pre->visit(x) will return the expression in prefix format and x->accept(pos) or pos->visit(x) will return the expression in postfix format.

Be careful that you are not allowed to use **any function to check the type of an object** when implementing this exercise

Tip: Use Visitor pattern.

Answer: (penalty regime: 0 %)

```
#include <iostream>
using namespace std;

class Eval;
class PrintPrefix;
class PrintPostfix;

class VisitorInt;
class VisitorVoid;
class BinExp;
class UnExp;
class IntLit;

class Exp{
    public:
        int v;
        virtual int accept(VisitorInt* ) = 0;
        virtual void accept(VisitorVoid* ) = 0;
};


class VisitorVoid{
```

	Test	Expected	Got	
✔	`cout << eval->visit(bii2);`	`6`	`6`	✔
✔	`cout << eval->visit(i2);`	`2`	`2`	✔
✔	`cout << eval->visit(bii1);`	`3`	`3`	✔
✔	`cout << eval->visit(bii3);`	`5`	`5`	✔
✔	`cout << eval->visit(bii4);`	`30`	`30`	✔
✔	`cout << bii5->accept(eval);`	`10`	`10`	✔
✔	`cout << eval->visit(u1);`	`-6`	`-6`	✔
✔	`pre->visit(i1);`	`1`	`1`	✔
✔	`pre->visit(i2);`	`2`	`2`	✔
✔	`pre->visit(bii1);`	`+ 1 2`	`+ 1 2`	✔
✔	`pre->visit(bii2);`	`* + 1 2 2`	`* + 1 2 2`	✔
✔	`pre->visit(bii3);`	`- * + 1 2 2 1`	`- * + 1 2 2 1`	✔
✔	`pre->visit(bii4);`	`* * + 1 2 2 - * + 1 2 2 1`	`* * + 1 2 2 - * + 1 2 2 1`	✔
✔	`pre->visit(bii5);`	`/ * * + 1 2 2 - * + 1 2 2 1 + 1 2`	`/ * * + 1 2 2 - * + 1 2 2 1 + 1 2`	✔
✔	`pre->visit(u1);`	`-. * + 1 2 2`	`-. * + 1 2 2`	✔
✔	`pos->visit(bii2);`	`1 2 + 2 *`	`1 2 + 2 *`	✔
✔	`pos->visit(bii3);`	`1 2 + 2 * 1 -`	`1 2 + 2 * 1 -`	✔
✔	`pos->visit(u1);`	`1 2 + 2 * -.`	`1 2 + 2 * -.`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.