컴퓨터 프로그래밍 2
-hw11-

학번 : 201602038
제출일 : 2017.5.29.
이름 : 이 미 진

1. 함수 설명

1-1. 무순 리스트의 성능 측정

1) AppIO

1-1) AppIO* AppIO_new(void): AppIO 객체 생성

1-2) void AppIO_delete(AppIO* _this): 주어진 AppIO 객체 소멸시킴

1-3) void AppIO_out_message(AppIO* _this, char* aMessage): 주어진 메시지 문자열 aMessage출력

2) AppController

2-1) AppController* AppController_new(void): AppController 객체 생성

2-2) void AppController_delete(AppController* _this): AppController 객체 소멸

2-3) void AppController_run(AppController* _this): App 실행

2-4) void AppController_generateTestDataByRandomNumbers(AppController* _this, UnsortedLinkedList* aList, int aMaxSize): 랜덤값 생성

2-5) long AppController_timeForUnsortedLinkedList_add(AppController* _this, UnsortedLinkedList* aList, Element aTestData[], int aTestSize): 삽입 시간

2-6) long AppController_timeForUnsortedLinkedList_removeMax(AppController* _this, UnsortedLinkedList* aList, int aTestSize): 최대값 삭제하는 시간

2-7) long AppController_timeForUnsortedvList_min (AppController* _this,UnsortedLinkedList* aList, int aTestSize): 최소값 찾는 시간

2-8) void AppController_showResults(AppController* _this, int aTestSize, long aTimeForAdd,long aTimeForMin,long aTimeForRemoveMax): 결과값 출력

3) UnsortedLinkedList

4-1) UnsortedLinkedList* UnsortedLinkedList_new(): 객체 생성

4-2) void UnsortedLinkedList_delete(UnsortedLinkedList* _this):객체 소멸

4-3) Boolean UnsortedLinkedList_isEmpty (UnsortedLinkedList* _this): 비었는지 검사

4-4) Boolean UnsortedLinkedList_isFull (UnsortedLinkedList* _this): 꽉 찼는지 검사

4-5) Boolean UnsortedLinkedList_add (UnsortedLinkedList* _this,Element anElement): 삽입

4-6) Element UnsortedLinkedList_min (UnsortedLinkedList* _this): 최소값 찾기

4-7) Element UnsortedLinkedList_removeMax(UnsortedLinkedList* _this): 최대값 삭제

1-2. 순서 리스트의 성능 측정

1) AppIO

1-1) AppIO* AppIO_new(void): AppIO 객체 생성

1-2) void AppIO_delete(AppIO* _this): 주어진 AppIO 객체 소멸시킴

1-3) void AppIO_out_message(AppIO* _this, char* aMessage): 주어진 메시지 문자열 aMessage출력

2) AppController

2-1) AppController* AppController_new(void): AppController 객체 생성

2-2) void AppController_delete(AppController* _this):AppController 객체 소멸

2-3) void AppController_run(AppController* _this): App 실행

2-4) void AppController_generateTestDataByRandomNumbers(AppController* _this,int aTestData[], int aMaxSize) : 랜덤값 생성

2-5) long AppController_timeForUnsortedLinkedList_add(AppController* _this, UnsortedLinkedList* uList, Element aTestData[], int aTestSize) : 무순 리스트의 삽입 측정 시간

2-6) long AppController_timeForUnsortedLinkedList_removeMax(AppController* _this, UnsortedLinkedList* uList, int aTestSize) : 무순 리스트의 최대값 삭제 측정 시간

2-7) long AppController_timeForUnsortedLinkedList_min (AppController* _this,UnsortedLinkedList* uList, int aTestSize) : 무순 리스트의 최소값 찾기 측정 시간

2-8) long AppController_timeForSortedLinkedList_add(AppController* _this,
2-9) SortedLinkedList* sList, Element aTestData[], int aTestSize) : 순서 리스트의 삽입 측정 시간

2-10) long AppController_timeForSortedLinkedList_removeMax(AppController* _this, SortedLinkedList* sList, int aTestSize) : 순서 리스트의 최대값 삭제 측정 시간

2-11) long AppController_timeForSortedLinkedList_min (AppController* _this,SortedLinkedList* sList, int aTestSize) : 순서 리스트의 최소값 찾기 측정 시간

2-12) void AppController_showResults(AppController* _this, int aTestSize, long aTimeForAdd,long aTimeForMin,long aTimeForRemoveMax) : 성능 측정 결과 값을 보여줌

3) UnsortedLinkedList

3-1) UnsortedLinkedList* UnsortedLinkedList_new(): 객체 생성

3-2) void UnsortedLinkedList_delete(UnsortedLinkedList* _this): 객체 소멸

3-3) Boolean UnsortedLinkedList_isEmpty (UnsortedLinkedList* _this): 무순 리스트가 비었는지 검사

3-4) Boolean UnsortedLinkedList_isFull (UnsortedLinkedList* _this): 무순 리스트가 꽉 찼는지 검사

3-5) Boolean UnsortedLinkedList_add (UnsortedLinkedList* _this,Element anElement): 삽입

3-6) Element UnsortedLinkedList_min (UnsortedLinkedList* _this): 최소값 찾기

3-7) Element UnsortedLinkedList_removeMax(UnsortedLinkedList* _this): 최대값 삭제

4) SortedLinkedList

4-1) SortedLinkedList* SortedLinkedList_new(): 순서 리스트 객체 생성

4-2) void SortedLinkedList_delete(SortedLinkedList* _this): 순서 리스트 객체 소멸

4-3) Boolean SortedLinkedList_isEmpty(SortedLinkedList* _this): 순서 리스트가 비었는지 검사

4-4) Boolean SortedLinkedList_isFull(SortedLinkedList* _this): 순서 리스트가 꽉 찼는지 검사

4-5) Boolean SortedLinkedList_add(SortedLinkedList* _this, Element anElement): 순서 리스트에 삽입

4-6) Element SortedLinkedList_min(SortedLinkedList* _this): 순서 리스트에서 최소값 찾기

4-7) Element SortedLinkedList_removeMax(SortedLinkedList* _this): 순서 리스트에서 최대값 삭제

2. 전체 코드

2-1. 무순 리스트의 성능 측정

1) main.c

```c
//
//  main.c
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include <stdio.h>
#include "AppController.h"

int main(void) {
    AppController* appController = AppController_new(); //
appController 객체 생성
    AppController_run(appController);   // 프로그램 실행
    AppController_delete(appController); // appController 객체 소멸

    return 0;
}
```

2) AppIO.c

```c
//
//  AppIO.c
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"

struct _AppIO {

};

AppIO* AppIO_new(void) {    //AppIO 객체 생성
    AppIO* _this = NewObject(AppIO);

    return _this;
}

void AppIO_delete(AppIO* _this) {   // AppIO 객체 소멸
    free(_this);
}
```

```
void AppIO_out_message(AppIO* _this,char* aMessage) {   // 메시지 출
력
    printf("%s",aMessage);
}
```

3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

#include <stdio.h>
#include "Message.h"
#include "Common.h"

typedef struct _AppIO   AppIO;

AppIO* AppIO_new(void);    // AppIO 객체 생성
void AppIO_delete(AppIO* _this);   // 주어진 AppIO 객체 소멸시킴
void AppIO_out_message(AppIO* _this, char* aMessage);  // 주어진 메시
지 문자열 aMessage출력


#endif /* AppIO_h */
```

4) AppController.c

```
//
//  AppController.c
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppController.h"
#include "Common.h"
#include "Message.h"
#include "AppIO.h"
#include "Timer.h"
#include <time.h>
```

```c
#include "UnsortedLinkedList.h"
#include "SortedLinkedList.h"

#define MIN_TEST_SIZE   1000
#define NUMBER_OF_TESTS 5
#define TEST_SIZE_INTERVAL  1000
#define MAX_DATA_SIZE   MIN_TEST_SIZE +
(TEST_SIZE_INTERVAL*(NUMBER_OF_TESTS-1))

// 비공개 함수
void AppController_generateTestDataByRandomNumbers(AppController*
_this,int aTestData[], int aMaxSize);  // 랜덤값 생성

long AppController_timeForUnsortedLinkedList_add(AppController*
_this, UnsortedLinkedList* aList, Element aTestData[], int
aTestSize);  // 삽입 시간
long
AppController_timeForUnsortedLinkedList_removeMax(AppController*
_this, UnsortedLinkedList* aList, int aTestSize); // 최대값 삭제하는 시
간
long AppController_timeForUnsortedvList_min (AppController*
_this,UnsortedLinkedList* aList, int aTestSize);    // 최소값 찾는 시
간

void AppController_showResults(AppController* _this, int
aTestSize, long aTimeForAdd,long aTimeForMin,long
aTimeForRemoveMax);    // 결과값 출력

struct _AppController {
    AppIO* _appIO;
};

AppController* AppController_new(void) {    // AppController 객체 생
성
    AppController* _this = NewObject(AppController);
    _this->_appIO = AppIO_new();
    return _this;
}

void AppController_delete(AppController* _this) {   //
AppController 객체 소멸
    AppIO_delete(_this->_appIO);
    free(_this);
}

void AppController_generateTestDataByRandomNumbers(AppController*
_this,int aTestData[], int aMaxSize){    // 난수 생성
    srand((unsigned)time(NULL));
    for(int i = 0; i < aMaxSize; i++){
        aTestData[i] = rand();
```

```
    }
}

void AppController_showResults(AppController* _this, int
aTestSize, long aTimeForAdd,long aTimeForMin, long
aTimeForRemoveMax) {    // 결과값 출력
    char results[255];
    sprintf(results, "크기 : %4d, 삽입 : %ld, 최소값얻기 : %ld, 최대값삭제
: %ld\n", aTestSize, aTimeForAdd, aTimeForMin, aTimeForRemoveMax);
    AppIO_out_message(_this->_appIO, results);
}

long AppController_timeForUnsortedLinkedList_add(AppController*
_this, UnsortedLinkedList* aList, Element aTestData[], int
aTestSize) { // 삽입 시간
    long duration = 0;
    int i;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !UnsortedLinkedList_isFull(aList)) {
            UnsortedLinkedList_add(aList, aTestData[i]);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;

}

long
AppController_timeForUnsortedLinkedList_removeMax(AppController*
_this, UnsortedLinkedList* aList, int aTestSize){ // 최대값 삭제 시간
    long duration = 0;
    int i;
    Element max;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !UnsortedLinkedList_isEmpty(aList)) {
            max = UnsortedLinkedList_min(aList);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);
```

```
    return duration;
}

long AppController_timeForUnsortedvList_min (AppController*
_this,UnsortedLinkedList* aList, int aTestSize){    // 최소값 찾는 시
간
    long duration = 0;
    int i;
    Element min;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !UnsortedLinkedList_isEmpty(aList)) {
            min = UnsortedLinkedList_min(aList);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;
}

void AppController_run(AppController* _this) {  // 실행
    int i;
    int testData[MAX_DATA_SIZE];
    int testSize = MIN_TEST_SIZE;
    double timeForAdd,timeForMin,timeForRemoveMax;

    AppIO_out_message(_this->_appIO,
MSG_StartPerformanceMeasuring);
    AppIO_out_message(_this->_appIO,
MSG_TitleForUnsortedArrayList);

    AppController_generateTestDataByRandomNumbers(_this,testData,
MAX_DATA_SIZE);

    for( i=0; i<NUMBER_OF_TESTS; i++ ) {
        UnsortedLinkedList* aList = UnsortedLinkedList_new();
        timeForAdd =
AppController_timeForUnsortedLinkedList_add(_this, aList,
testData, testSize);
        timeForMin = AppController_timeForUnsortedvList_min(_this,
aList, testSize);
        timeForRemoveMax =
AppController_timeForUnsortedLinkedList_removeMax(_this, aList,
testSize);

        AppController_showResults(_this, testSize, timeForAdd,
timeForMin, timeForRemoveMax);
```

```
        UnsortedLinkedList_delete(aList);
        testSize += TEST_SIZE_INTERVAL;
    }
    AppIO_out_message(_this->_appIO, MSG_EndPerformanceMeasuring);
}
```

5) AppController.h

```
//
//  AppController.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

typedef struct _AppController AppController;

AppController* AppController_new(void);    // AppController 객체 생
성
void AppController_delete(AppController* _this);   //
AppController 객체 소멸
void AppController_run(AppController* _this);  // App 실행

#endif /* AppController_h */
```

6) Common.h

```
//
//  Common.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

#include<stdlib.h>

typedef int Element ;

#define NewVector(TYPE,SIZE) (TYPE*)malloc(sizeof(TYPE)*SIZE)
#define NewObject(TYPE) (TYPE*)malloc(sizeof(TYPE))
```

```
typedef enum {FALSE,TRUE} Boolean;  // FALSE와 TRUE 값을 갖는 Boolean
선언


#endif /* Common_h */
```

7) UnsortedLinkedList.c

```
//
//  UnsortedLinkedList.c
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "UnsortedLinkedList.h"

struct _UnsortedLinkedList {
    int _size;
    Node* _head;
};

UnsortedLinkedList* UnsortedLinkedList_new() {  //
UnsortedArrayList 객체 생성
    UnsortedLinkedList* _this = NewObject(UnsortedLinkedList);
    _this->_size = 0;
    _this->_head = NULL;
    return _this;
}

void UnsortedLinkedList_delete (UnsortedLinkedList* _this) {  //
UnsortedArrayList 객체 소멸
    free(_this);
}

//배열이 비었을 경우
Boolean UnsortedLinkedList_isEmpty (UnsortedLinkedList* _this) {
// 무순 리스트가 비었는지 검사
    return ( _this->_size == 0 );
}

// 배열이 꽉찼을 경우
Boolean UnsortedLinkedList_isFull (UnsortedLinkedList* _this) { //
무순 리스트가 꽉 찼는지 검사
    return FALSE;
}
```

```
Boolean UnsortedLinkedList_add (UnsortedLinkedList* _this, Element
anElement) { // 무순 리스트에 삽입
    Node* addedNode;
    addedNode = Node_new();
    addedNode->_element = anElement;
    addedNode->_next = _this->_head;
    _this->_head = addedNode;
    _this->_size++;

    return TRUE;
}

Element UnsortedLinkedList_min (UnsortedLinkedList* _this) {
    int min = _this->_head->_element;

    Node* currentNode = _this->_head->_next;
    while( currentNode != NULL ) {
        if( min > currentNode->_element ) {
            min = currentNode->_element;
        }
        currentNode = currentNode->_next;
    }

    return min;
}

Element UnsortedLinkedList_removeMax(UnsortedLinkedList* _this) {
// 최대값 삭제
    int Max = _this->_head->_element;
    int removeElement;
    Node* removeMax = NULL;
    Node* preNode = _this->_head;
    Node* currentNode = _this->_head->_next;

    while( currentNode != NULL ) {
        if( Max < currentNode->_element ) {
            Max = currentNode->_element;
            removeMax = currentNode;
        }
        currentNode = currentNode->_next;
        preNode = preNode->_next;
    }

    removeElement = removeMax->_element;

    return removeElement;
}
```

8) UnsortedLinkedList.h

```
//
//  UnsortedLinkedList.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef UnsortedLinkedList_h
#define UnsortedLinkedList_h

#include <stdio.h>
#include "Common.h"
#include "List.h"
#include "Node.h"

typedef struct _UnsortedLinkedList  UnsortedLinkedList;

UnsortedLinkedList* UnsortedLinkedList_new();   // 객체 생성
void UnsortedLinkedList_delete(UnsortedLinkedList* _this);  // 객체
소멸
Boolean UnsortedLinkedList_isEmpty (UnsortedLinkedList* _this); //
비었는지 검사
Boolean UnsortedLinkedList_isFull (UnsortedLinkedList* _this); //
꽉 찼는지 검사
Boolean UnsortedLinkedList_add (UnsortedLinkedList* _this,Element
anElement);   // 삽입
Element UnsortedLinkedList_min (UnsortedLinkedList* _this); // 최소
값 찾기
Element UnsortedLinkedList_removeMax(UnsortedLinkedList* _this);
// 최대값 삭제

#endif /* UnsortedLinkedList_h */
```

9) Node.c

```
//
//  Node.c
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Node.h"


Node* Node_new() {  // 노드 객체 생성
```

```
    Node* _this = NewObject(Node);
    _this->_element = 0;
    _this->_next = NULL;
    return _this;
}

void Node_delete(Node* _this){  // 노드 객체 소멸
    free(_this);
}
```

10) Node.h

```
//
//  Node.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Node_h
#define Node_h

#include <stdio.h>
#include "Common.h"

typedef struct _Node Node;
struct _Node {
    Element _element;
    Node* _next;
};

Node* Node_new();   // 노드 객체 생성
void Node_delete(Node* _this);  // 노드 객체 소멸

#endif /* Node_h */
```

11) Message.h

```
//
//  Message.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Message_h
#define Message_h
```

```
#define MSG_StartPerformanceMeasuring   "<성능 측정을 시작합니다>\n"
#define MSG_EndPerformanceMeasuring "<성능 측정을 종료합니다>\n"
#define MSG_TitleForUnsortedArrayList "[Unsorted Linked List]\n"

#endif /* Message_h */
```

12) Timer.c

```
//
//  Timer.c
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Timer.h"
#include "Common.h"
#include <math.h>


Timer* Timer_new(){//타이머 생성
    Timer* _this = NewObject(Timer);
    return _this;
}

void Timer_delete(Timer* _this){    //타이머 소멸
    free(_this);
}

void Timer_start(Timer* _this){ // 타이머 작동 시작
    _this->startCounter = clock();  // 실행 전 카운터 값을 얻음
}

void Timer_stop(Timer* _this){  //타이머 작동 중지
    _this->stopCounter = clock();//실행 후 카운터 값을 얻음
}

long Timer_duration(Timer* _this){//타이머 작동 시작부터 중지까지의 시간
    return (long)(_this->stopCounter - _this-
>startCounter)*(long)1000000 / (long)CLOCKS_PER_SEC;
}
```

13) Timer.h

```
//
//  Timer.h
//  CP2_WEEK11
//
//  Created by stu2017s10 on 2017. 5. 23..
```

```
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Timer_h
#define Timer_h

#include <stdio.h>
#include <time.h>

typedef struct{
    clock_t startCounter;   // clock_t 는 cpu clock수를 저장하는 구조체
    clock_t stopCounter;
} Timer;

Timer* Timer_new(); // 타이머 생성
void Timer_delete(Timer* _this);    // 타이머 소멸
void Timer_start(Timer* _this); // 타이머 작동 시작
void Timer_stop(Timer* _this);  // 타이머 작동 중지
long Timer_duration(Timer* _this);  // 타이머 작동 시작부터 중지까지의 시간


#endif /* Timer_h */
```

2-2. 순서 리스트의 성능 측정

1) main.c

```
//
//  main.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include <stdio.h>
#include "AppController.h"

int main(void) {
    AppController* appController = AppController_new(); //
appController 객체 생성
    AppController_run(appController);   // 프로그램 실행
    AppController_delete(appController); // appController 객체 소멸
```

```
    return 0;
}
```

2) AppIO.c

```
//
//  AppIO.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppIO.h"

struct _AppIO {

};

AppIO* AppIO_new(void) {    // AppIO 객체 생성
    AppIO* _this = NewObject(AppIO);

    return _this;
}

void AppIO_delete(AppIO* _this) {  // 주어진 AppIO 객체 소멸시킴
    free(_this);
}

void AppIO_out_message(AppIO* _this,char* aMessage) {  // 주어진 메시지 문자열 aMessage출력
    printf("%s",aMessage);
}
```

3) AppIO.h

```
//
//  AppIO.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppIO_h
#define AppIO_h

#include <stdio.h>
#include "Message.h"
```

```
#include "Common.h"

typedef struct _AppIO   AppIO;

AppIO* AppIO_new(void);    // AppIO 객체 생성
void AppIO_delete(AppIO* _this);   // 주어진 AppIO 객체 소멸시킴
void AppIO_out_message(AppIO* _this, char* aMessage);  // 주어진 메시지 문자열 aMessage출력


#endif /* AppIO_h */
```

4) AppController.c

```
//
//  AppController.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "AppController.h"
#include "Common.h"
#include "Message.h"
#include "AppIO.h"
#include "Timer.h"
#include <time.h>
#include "UnsortedLinkedList.h"
#include "SortedLinkedList.h"

#define MIN_TEST_SIZE   10000
#define NUMBER_OF_TESTS 5
#define TEST_SIZE_INTERVAL  10000
#define MAX_DATA_SIZE   MIN_TEST_SIZE +
(TEST_SIZE_INTERVAL*(NUMBER_OF_TESTS-1))

// 비공개 함수
void AppController_generateTestDataByRandomNumbers(AppController*
_this,int aTestData[], int aMaxSize); // 랜덤값 생성성
long AppController_timeForUnsortedLinkedList_add(AppController*
_this, UnsortedLinkedList* uList, Element aTestData[], int
aTestSize);  // 무순 리스트의 삽입 측정 시간
long
AppController_timeForUnsortedLinkedList_removeMax(AppController*
_this, UnsortedLinkedList* uList, int aTestSize); // 무순 리스트의 최대
값 삭제 측정 시간
```

```
long AppController_timeForUnsortedLinkedList_min (AppController*
_this,UnsortedLinkedList* uList, int aTestSize);   // 무순 리스트의 최
소값 찾기 측정 시간

long AppController_timeForSortedLinkedList_add(AppController*
_this, SortedLinkedList* sList, Element aTestData[], int
aTestSize);  // 순서 리스트의 삽입 측정 시간
long
AppController_timeForSortedLinkedList_removeMax(AppController*
_this, SortedLinkedList* sList, int aTestSize); // 순서 리스트의 최대값
삭제 측정 시간
long AppController_timeForSortedLinkedList_min (AppController*
_this,SortedLinkedList* sList, int aTestSize);   // 순서 리스트의 최소
값 찾기 측정 시간

void AppController_showResults(AppController* _this, int
aTestSize, long aTimeForAdd,long aTimeForMin,long
aTimeForRemoveMax);  // 성능 측정 결과 값을 보여줌

struct _AppController {
    AppIO* _appIO;
};

AppController* AppController_new(void) {    // AppController 객체 생
성
    AppController* _this = NewObject(AppController);
    _this->_appIO = AppIO_new();
    return _this;
}

void AppController_delete(AppController* _this) {   //
AppController 객체 소멸
    AppIO_delete(_this->_appIO);
    free(_this);
}

void AppController_generateTestDataByRandomNumbers(AppController*
_this,int aTestData[], int aMaxSize){    // 난수 생성
    srand((unsigned)time(NULL));
    for(int i = 0; i < aMaxSize; i++){
        aTestData[i] = rand();
    }
}

void AppController_showResults(AppController* _this, int
aTestSize, long aTimeForAdd,long aTimeForMin, long
aTimeForRemoveMax) {
    char results[255];
    sprintf(results, "크기 : %4d, 삽입 : %ld, 최소값얻기 : %ld, 최대값삭제
: %ld\n", aTestSize, aTimeForAdd, aTimeForMin, aTimeForRemoveMax);
```

```
    AppIO_out_message(_this->_appIO, results);
}

long AppController_timeForUnsortedLinkedList_add(AppController*
_this, UnsortedLinkedList* uList, Element aTestData[], int
aTestSize) {
    long duration = 0;
    int i;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !UnsortedLinkedList_isFull(uList)) {
            UnsortedLinkedList_add(uList, aTestData[i]);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;

}

long
AppController_timeForUnsortedLinkedList_removeMax(AppController*
_this, UnsortedLinkedList* uList, int aTestSize){
    long duration = 0;
    int i;
    Element max;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !UnsortedLinkedList_isEmpty(uList)) {
            max = UnsortedLinkedList_min(uList);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;
}

long AppController_timeForUnsortedLinkedList_min (AppController*
_this,UnsortedLinkedList* uList, int aTestSize){
    long duration = 0;
    int i;
    Element min;
    Timer* timer = Timer_new();
```

```
    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !UnsortedLinkedList_isEmpty(uList)) {
            min = UnsortedLinkedList_min(uList);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;
}

long AppController_timeForSortedLinkedList_add(AppController*
_this, SortedLinkedList* sList, Element aTestData[], int
aTestSize){
    long duration = 0;
    int i;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !SortedLinkedList_isFull(sList)) {
            SortedLinkedList_add(sList, aTestData[i]);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;
}

long
AppController_timeForSortedLinkedList_removeMax(AppController*
_this, SortedLinkedList* sList, int aTestSize){
    long duration = 0;
    int i;
    Element max;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !SortedLinkedList_isEmpty(sList)) {
            max = SortedLinkedList_min(sList);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);
```

```
    return duration;
}

long AppController_timeForSortedLinkedList_min (AppController*
_this,SortedLinkedList* sList, int aTestSize){
    long duration = 0;
    int i;
    Element min;
    Timer* timer = Timer_new();

    for( i=0; i<aTestSize; i++ ) {
        Timer_start(timer);
        if( !SortedLinkedList_isEmpty(sList)) {
            min = SortedLinkedList_min(sList);
        }
        Timer_stop(timer);
        duration += Timer_duration(timer);
    }
    Timer_delete(timer);

    return duration;
}

void AppController_run(AppController* _this) {
    int i;
    int testData[MAX_DATA_SIZE];
    int testSize = MIN_TEST_SIZE;
    double timeForAdd,timeForMin,timeForRemoveMax;

    AppIO_out_message(_this->_appIO,
MSG_StartPerformanceMeasuring);
    AppController_generateTestDataByRandomNumbers(_this, testData,
MAX_DATA_SIZE);
    AppIO_out_message(_this->_appIO,
MSG_TitleForUnsortedArrayList);

    for( i=0; i<NUMBER_OF_TESTS; i++ ) {
        UnsortedLinkedList* uList = UnsortedLinkedList_new();
        timeForAdd =
AppController_timeForUnsortedLinkedList_add(_this, uList,
testData, testSize);
        timeForMin =
AppController_timeForUnsortedLinkedList_min(_this, uList,
testSize);
        timeForRemoveMax =
AppController_timeForUnsortedLinkedList_removeMax(_this, uList,
testSize);

        AppController_showResults(_this, testSize, timeForAdd,
timeForMin, timeForRemoveMax);
```

```
        UnsortedLinkedList_delete(uList);
        testSize += TEST_SIZE_INTERVAL;
    }

    testSize = MIN_TEST_SIZE;

    AppIO_out_message(_this->_appIO, MSG_TitleForSortedArrayList);

    for( i=0; i<NUMBER_OF_TESTS; i++ ) {
        SortedLinkedList* sList = SortedLinkedList_new();
        timeForAdd =
AppController_timeForSortedLinkedList_add(_this, sList, testData,
testSize);
        timeForMin =
AppController_timeForSortedLinkedList_min(_this, sList, testSize);
        timeForRemoveMax =
AppController_timeForSortedLinkedList_removeMax(_this, sList,
testSize);

        AppController_showResults(_this, testSize, timeForAdd,
timeForMin, timeForRemoveMax);
        SortedLinkedList_delete(sList);
        testSize += TEST_SIZE_INTERVAL;
    }

    AppIO_out_message(_this->_appIO, MSG_EndPerformanceMeasuring);

}
```

5) AppController.h

```
//
//  AppController.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef AppController_h
#define AppController_h

typedef struct _AppController AppController;

AppController* AppController_new(void);    // AppController 객체 생
성
void AppController_delete(AppController* _this);   //
AppController 객체 소멸
void AppController_run(AppController* _this);  // App 실행
```

```
#endif /* AppController_h */
```

6) Node.c

```
//
//  Node.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Node.h"

Node* Node_new() {  // 노드 삽입
    Node* _this = NewObject(Node);
    _this->_element = 0;
    _this->_next = NULL;
    return _this;
}

void Node_delete(Node* _this){  // 노드 삭제
    free(_this);
}
```

7) Node.h

```
//
//  Node.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Node_h
#define Node_h

#include <stdio.h>
#include "Common.h"

typedef struct _Node Node;
struct _Node {
    Element _element;
    Node* _next;
};

Node* Node_new();   // 노드 객체 생성
void Node_delete(Node* _this);  // 노드 객체 소멸
```

```
#endif /* Node_h */
```

8) UnsortedLinkedList.c

```
//
//  UnsortedLinkedList.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "UnsortedLinkedList.h"

struct _UnsortedLinkedList {
    int _size;
    Node* _head;
};

UnsortedLinkedList* UnsortedLinkedList_new() {  //
UnsortedArrayList 객체 생성
    UnsortedLinkedList* _this = NewObject(UnsortedLinkedList);
    _this->_size = 0;
    _this->_head = NULL;
    return _this;
}

void UnsortedLinkedList_delete (UnsortedLinkedList* _this) {  //
UnsortedArrayList 객체 소멸
    free(_this);
}

//배열이 비었을 경우
Boolean UnsortedLinkedList_isEmpty (UnsortedLinkedList* _this) {
// 무순 리스트가 비었는지 검사
    return ( _this->_size == 0 );
}

// 배열이 꽉찼을 경우
Boolean UnsortedLinkedList_isFull (UnsortedLinkedList* _this) { //
무순 리스트가 꽉 찼는지 검사
    return FALSE;
}

Boolean UnsortedLinkedList_add (UnsortedLinkedList* _this, Element
anElement) { // 무순 리스트에 삽입
    Node* addedNode;
    addedNode = Node_new();
    addedNode->_element = anElement;
```

```
    addedNode->_next = _this->_head;
    _this->_head = addedNode;
    _this->_size++;

    return TRUE;
}

Element UnsortedLinkedList_min (UnsortedLinkedList* _this) {    //
최소값 찾기
    int min = _this->_head->_element;

    Node* currentNode = _this->_head->_next;
    while( currentNode != NULL ) {
        if( min > currentNode->_element ) {
            min = currentNode->_element;
        }
        currentNode = currentNode->_next;
    }

    return min;
}

Element UnsortedLinkedList_removeMax(UnsortedLinkedList* _this) {
// 최대값 삭제
    int Max = _this->_head->_element;
    int removeElement;
    Node* removeMax = NULL;
    Node* preNode = _this->_head;
    Node* currentNode = _this->_head->_next;

    while( currentNode != NULL ) {
        if( Max < currentNode->_element ) {
            Max = currentNode->_element;
            removeMax = currentNode;
        }
        currentNode = currentNode->_next;
        preNode = preNode->_next;
    }

    removeElement = removeMax->_element;

    return removeElement;
}
```

9) UnsortedLinkedList.h

```
//
//  UnsortedLinkedList.h
//  CP2_WEEK11_2
//
```

```
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef UnsortedLinkedList_h
#define UnsortedLinkedList_h

#include <stdio.h>
#include "Common.h"
#include "Node.h"

typedef struct _UnsortedLinkedList  UnsortedLinkedList;

UnsortedLinkedList* UnsortedLinkedList_new();   // 객체 생성
void UnsortedLinkedList_delete(UnsortedLinkedList* _this);  // 객체
소멸
Boolean UnsortedLinkedList_isEmpty (UnsortedLinkedList* _this); //
무순 리스트가 비었는지 검사
Boolean UnsortedLinkedList_isFull (UnsortedLinkedList* _this);  //
무순 리스트가 꽉 찼는지 검사
Boolean UnsortedLinkedList_add (UnsortedLinkedList* _this,Element
anElement);   // 삽입
Element UnsortedLinkedList_min (UnsortedLinkedList* _this); // 최소
값 찾기
Element UnsortedLinkedList_removeMax(UnsortedLinkedList* _this);
// 최대값 삭제

#endif /* UnsortedLinkedList_h */
```

10) SortedLinkedList.c

```
//
//  SortedLinkedList.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "SortedLinkedList.h"

struct _SortedLinkedList {
    int _size;
    Node* _head;
};

SortedLinkedList* SortedLinkedList_new() {   // 순서 리스트 객체 생성
    SortedLinkedList* _this = NewObject(SortedLinkedList);
    _this->_size = 0;
```

```
    _this->_head = NULL;

    return _this;
}

void SortedLinkedList_delete(SortedLinkedList* _this){  // 순서 리스
트 객체 소멸
    free(_this);
}

Boolean SortedLinkedList_isEmpty(SortedLinkedList* _this){  // 순서
리스트가 비었는지 검사
    return (_this->_size == 0);
}

Boolean SortedLinkedList_isFull(SortedLinkedList* _this){   // 순서
리스트가 꽉 찼는지 검사
    return FALSE;
}

Boolean SortedLinkedList_add(SortedLinkedList* _this, Element
anElement) {   // 순서 리스트에 삽입
    Node* currentNode = _this->_head;
    Node* addedNode = Node_new();
    addedNode->_element = anElement;
    addedNode->_next = NULL;

    if( _this->_head == NULL ) {
        _this->_head = addedNode;
        _this->_size++;
    }
    else if( addedNode->_element <= currentNode->_element  ) {
        addedNode->_next = currentNode;
        _this->_head = addedNode;
        _this->_size++;
    }
    else {
        while( (currentNode->_next != NULL) && ( addedNode-
>_element <= currentNode->_next->_element) ) {
            currentNode = currentNode->_next;
        }
        addedNode->_next = currentNode->_next;
        currentNode->_next = addedNode;
        _this->_size++;
    }
    return TRUE;

}
```

```
Element SortedLinkedList_min(SortedLinkedList* _this) {  // 순서 리스트에서 최소값 찾기
    return _this->_head->_element;
}

Element SortedLinkedList_removeMax(SortedLinkedList* _this)
{    // 순서 리스트에서 최대값 삭제
    Node* currentNode = _this->_head;
    Element removedElement;

    while ( currentNode->_next != NULL ) {
        currentNode = currentNode->_next;
    }
    removedElement = currentNode->_element;
    currentNode = NULL;

    return removedElement;
}
```

11) SortedLinkedList.h

```
//
//  SortedLinkedList.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef SortedLinkedList_h
#define SortedLinkedList_h

#include <stdio.h>
#include "Common.h"
#include "Node.h"

typedef struct _SortedLinkedList SortedLinkedList;

SortedLinkedList* SortedLinkedList_new();   // 순서 리스트 객체 생성
void SortedLinkedList_delete(SortedLinkedList* _this);  // 순서 리스트 객체 소멸
Boolean SortedLinkedList_isEmpty(SortedLinkedList* _this);  // 순서 리스트가 비었는지 검사
Boolean SortedLinkedList_isFull(SortedLinkedList* _this);   // 순서 리스트가 꽉 찼는지 검사
Boolean SortedLinkedList_add(SortedLinkedList* _this, Element anElement);   // 순서 리스트에 삽입
Element SortedLinkedList_min(SortedLinkedList* _this);  // 순서 리스트에서 최소값 찾기
```

```
Element SortedLinkedList_removeMax(SortedLinkedList* _this);    //
순서 리스트에서 최대값 삭제

#endif /* SortedLinkedList_h */
```

12) Common.h

```
//
//  Common.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Common_h
#define Common_h

#include<stdlib.h>

typedef int Element ;

#define NewVector(TYPE,SIZE) (TYPE*)malloc(sizeof(TYPE)*SIZE)
#define NewObject(TYPE) (TYPE*)malloc(sizeof(TYPE))
typedef enum {FALSE,TRUE} Boolean;  // FALSE와 TRUE 값을 갖는 Boolean
선언

#endif /* Common_h */
```

13) Message.h

```
//
//  Message.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Message_h
#define Message_h

#define MSG_StartPerformanceMeasuring   "<성능 측정을 시작합니다>\n"
#define MSG_EndPerformanceMeasuring "<성능 측정을 종료합니다>\n"
#define MSG_TitleForUnsortedArrayList "[Unsorted Linked List]\n"
#define MSG_TitleForSortedArrayList "[Sorted Linked List]\n"

#endif /* Message_h */
```

14) Timer.c

```
//
//  Timer.c
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#include "Timer.h"
#include "Common.h"
#include <math.h>


Timer* Timer_new(){//타이머 생성
    Timer* _this = NewObject(Timer);
    return _this;
}

void Timer_delete(Timer* _this){    //타이머 소멸
    free(_this);
}

void Timer_start(Timer* _this){ // 타이머 작동 시작
    _this->startCounter = clock();  // 실행 전 카운터 값을 얻음
}

void Timer_stop(Timer* _this){  //타이머 작동 중지
    _this->stopCounter = clock();//실행 후 카운터 값을 얻음
}

long Timer_duration(Timer* _this){//타이머 작동 시작부터 중지까지의 시간
    return (long)(_this->stopCounter - _this-
>startCounter)*(long)1000000 / (long)CLOCKS_PER_SEC;
}
```

15) Timer.h

```
//
//  Timer.h
//  CP2_WEEK11_2
//
//  Created by stu2017s10 on 2017. 5. 29..
//  Copyright © 2017년 stu2017s10. All rights reserved.
//

#ifndef Timer_h
#define Timer_h
```

```c
#include <stdio.h>
#include <time.h>

typedef struct{
    clock_t startCounter;   // clock_t 는 cpu clock수를 저장하는 구조체
    clock_t stopCounter;
} Timer;

Timer* Timer_new(); // 타이머 생성
void Timer_delete(Timer* _this);    // 타이머 소멸
void Timer_start(Timer* _this); // 타이머 작동 시작
void Timer_stop(Timer* _this);  // 타이머 작동 중지
long Timer_duration(Timer* _this);  // 타이머 작동 시작부터 중지까지의 시간

#endif /* Timer_h */
```

3. 전체 설명

3-1. 무순 리스트의 성능 측정

1) main에서 AppController_new() 함수를 통해 appController 객체를 생성한다.

2) AppController_run(appController) 함수로 프로그램을 실행하고, AppController_run() 에서는 AppIO_out_message(_this->_appIO, MSG_StartPerformanceMeasuring) 함수로 프로그램 시작 메시지를 출력한다.

3) AppController_generateTestDataByRandomNumbers() 함수로 랜덤값을 생성해 무작위 수에 대한 성능을 측정하도록 한다.

4) 우선, for문을 이용해 0부터 NUMBER_OF_TESTS (5) 까지 반복을 하는데 UnsortedLinkedList_new() 함수를 통해 무순 리스트 객체 aList를 생성한다.

5) AppController_timeForUnsortedLinkedList_add(), AppController_timeForUnsortedLinkedList_min(), AppController_timeForUnsortedLinkedList_removeMax() 함수를 통해 각각 삽입 측정 시간, 최솟값 찾기 측정 시간, 최댓값 삭제 측정 시간을 구하고 각 변수에 저장한다.

6) AppController_showResults() 를 통해 측정 결과 값을 출력한다.

7) 결과값이 출력되면 UnsortedLinkedList_delete(aList) 를 통해 aList 객체를 소멸시킨다.

8) 반복이 끝나면 성능 측정을 종료한다는 메시지를 출력하고 프로그램을 종료한다.

9) AppController_run(appController)의 실행이 끝나면 AppController_delete(appController) 함수가 appController객체를 소멸시킨다.

3-2. 순서 리스트의 성능 측정

1) main에서 AppController_new() 함수를 통해 appController 객체를 생성한다.

2) AppController_run(appController) 함수로 프로그램을 실행하고, AppController_run() 에서는 AppIO_out_message(_this->_appIO, MSG_StartPerformanceMeasuring) 함수로 프로그램 시작 메시지를 출력한다.

3) AppController_generateTestDataByRandomNumbers() 함수로 랜덤값을 생성해 무작위 수에 대한 성능을 측정하도록 한다.

4) 우선, for문을 이용해 0부터 NUMBER_OF_TESTS (5) 까지 반복을 하는데 UnsortedLinkedList_new() 함수를 통해 무순 리스트 객체 uList를 생성한다.

5) AppController_timeForUnsortedLinkedList_add(), AppController_timeForUnsortedLinkedList_min(), AppController_timeForUnsortedLinkedList_removeMax() 함수를 통해 각각 삽입 측정 시간, 최솟값 찾기 측정 시간, 최댓값 삭제 측정 시간을 구하고 각 변수에 저장한다.

6) AppController_showResults() 를 통해 측정 결과 값을 출력한다.

7) 결과값이 출력되면 UnsortedLinkedList_delete(uList) 를 통해 uList 객체를 소멸시킨다.

8) 무순 리스트에 대한 반복이 끝나면, testSize를 MIN_TEST_SIZE(10000)으로 초기화 시켜준다.

9) 순서 리스트 성능 측정 메시지를 출력하고, 다시 for문을 통해 0부터 NUMBER_OF_TESTS (5)까지 반복한다.

10) SortedLinkedList_new()를 통해 순서 리스트 객체 sList를 생성하고, AppController_timeForSortedLinkedList_add(),AppController_timeForSortedLinkedList_min(),AppController_timeForSortedLinkedList_removeMax() 함수를 통해 각각 삽입 측정 시간, 최솟값 찾기 측정 시간, 최댓값 삭제 측정 시간을 구하고 각 변수에 구한다.

11) AppController_showResults() 함수가 측정 결과값을 출력하고, SortedLinkedList_delete(sList) 함수가 sList 객체를 소멸시킨다.

12) 반복이 끝나면 성능 측정을 종료한다는 메시지를 출력하고 프로그램을 종료한다.

13) AppController_run(appController)의 실행이 끝나면 AppController_delete(appController) 함수가 appController객체를 소멸시킨다.

4. 실행 결과

4-1. 무순 리스트 성능 측정

```
<성능 측정을 시작합니다>
[Unsorted Linked List]
크기 : 1000, 삽입 : 483, 최소값얻기 : 4949, 최대값삭제 : 4231
크기 : 2000, 삽입 : 964, 최소값얻기 : 15863, 최대값삭제 : 15554
크기 : 3000, 삽입 : 1397, 최소값얻기 : 36524, 최대값삭제 : 36158
크기 : 4000, 삽입 : 1886, 최소값얻기 : 62736, 최대값삭제 : 60454
크기 : 5000, 삽입 : 2332, 최소값얻기 : 98796, 최대값삭제 : 94183
<성능 측정을 종료합니다>
Program ended with exit code: 0
```

4-2. 순서 리스트 성능 측정

```
<성능 측정을 시작합니다>
[Unsorted Linked List]
크기 : 10000, 삽입 : 4484, 최소값얻기 : 387916, 최대값삭제 : 383940
크기 : 20000, 삽입 : 8330, 최소값얻기 : 1514277, 최대값삭제 : 1508746
크기 : 30000, 삽입 : 11912, 최소값얻기 : 3385153, 최대값삭제 : 3375848
크기 : 40000, 삽입 : 15576, 최소값얻기 : 6009939, 최대값삭제 : 6004843
크기 : 50000, 삽입 : 19619, 최소값얻기 : 9381464, 최대값삭제 : 9367939
[Sorted Linked List]
크기 : 10000, 삽입 : 42740, 최소값얻기 : 3292, 최대값삭제 : 3799
크기 : 20000, 삽입 : 92262, 최소값얻기 : 6514, 최대값삭제 : 6572
크기 : 30000, 삽입 : 243744, 최소값얻기 : 9775, 최대값삭제 : 9867
크기 : 40000, 삽입 : 725289, 최소값얻기 : 13138, 최대값삭제 : 13654
크기 : 50000, 삽입 : 1822470, 최소값얻기 : 16531, 최대값삭제 : 16389
<성능 측정을 종료합니다>
Program ended with exit code: 0
```

5. 무순 리스트와 순서 리스트의 성능 분석

1) 삽입 시간 : 무순 리스트 < 순서 리스트
2) 최소값 얻는 시간 : 무순 리스트 > 순서 리스트
3) 최대값 삭제 시간 : 무순 리스트 > 순서 리스트