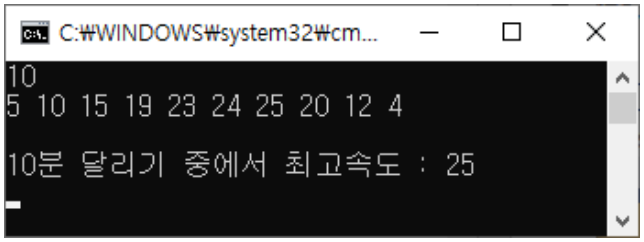
10장. 응용예제 답

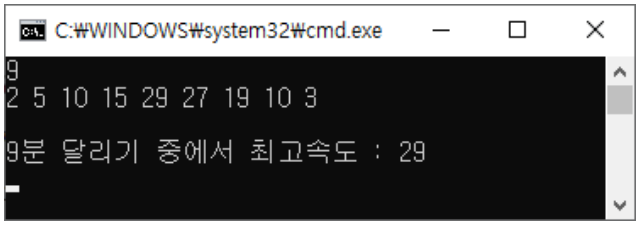
응용예제 15 러닝 머신의 최고 속도 구하기

topSpeed.c

```c
//응용예제16. 달리기 최고 속도 출력하기
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define MAX 60

int bitonic(int series[], int size, int start, int end);

int main(void) {
	int N, i, middle, topSpeed, history[MAX] = {0};

	//입력: 첫째 줄  => 달리기 총 시간 (분)
	scanf("%d", &N);
	//입력: 둘째 줄 => 분단위 속도 데이터
	for (i = 0; i < N; i++)
		scanf("%d", &history[i]);
	//최고 값 찾기 : 바이토닉 수열
	topSpeed = bitonic(history, N, 0, N - 1);

	printf("₩n%d분 달리기 중에서 최고속도 : %d₩n", N, topSpeed);

	getch(); return 0;
}

int bitonic(int series[], int size, int start, int end){
	int middle;
	if (size == 1) return series[start];
	if (size == 2) return ((series[start] > series[end]) ? series[start] : series[end]);

	middle = start + size / 2;
	if (series[middle] > series[middle + 1] && series[middle] > series[middle - 1])
		return series[middle];
	else if (series[middle] > series[middle + 1]) {
		bitonic(series, (middle - start), start, middle - 1);
	}
	else bitonic(series, (end - middle), middle + 1, end);
}
```

[실행 화면]




C:₩WINDOWS₩system32₩cmd.exe
9
2 5 10 15 29 27 19 10 3
9분 달리기 중에서 최고속도 : 29

응용예제 16

우편물 정리

체이닝 방법으로 해시 테이블을 구성하기 위해서, [예제 4-2] 단순연결리스트를 수정하여 사용.

hash.h

```
#pragma once
typedef char element;

// 버켓의 노드 (슬롯) 구조체 선언
typedef struct bucketNode {
    int key;
    element value;
    struct bucketNode* link;
} bucketNode;

// 버켓 구조체 선언
typedef struct bucket {
    bucketNode* head; // 버켓 가장 앞에 있는 노드의 포인터
    int count; // 버켓에 들어있는 노드의 개수
}bucket;

int hashFunction(int BUCKET_SIZE, int key);
bucketNode* createNode(int key, element value);
void addSlot(bucket* hashTable, int BUCKET_SIZE, int key, element value);
bucketNode* searchSlot(bucket* hashTable, int BUCKET_SIZE, int x);
void printHashTableAll(bucket* hashTable, int BUCKET_SIZE);
void printHashTable(bucket* hashTable, int BUCKET_SIZE, int key);
```

hash.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include "hash.h"

// 해쉬함수 : 제산함수
//사무실주소 숫자 중 백 자리는 층을 구분하므로, 하위 두자리(key%100)값에 대해 제산함수 적용
int hashFunction(int BUCKET_SIZE, int key) {
    return (key%100) % BUCKET_SIZE;
}
// 새 노드(슬롯) 생성 함수
bucketNode* createNode(int key, element value) {
    bucketNode* newNode;
    // 메모리 할당
    newNode = (bucketNode*)malloc(sizeof(bucketNode));
    // 사용자가 전해준 값을 대입
    newNode->key = key;
    newNode->value = value;
    newNode->link = NULL; // 생성할 때는 next를 NULL로 초기화
    return newNode;
}
```

```c
// 새로운 슬롯 추가: 입력 순서대로 구성하기 위해, 버킷리스트의 마지막 노드로 삽입
void addSlot(bucket* hashTable, int BUCKET_SIZE, int key, element value) {
    bucketNode* last;
    // 슬롯을 추가할 버켓 주소 계산 : 해싱함수
    int hashIndex = hashFunction(BUCKET_SIZE, key);
    // 새 노드 생성
    bucketNode* newNode = createNode(key, value);
    last = hashTable[hashIndex].head;
    // 버켓이 공백인 경우
    if (last == NULL) {
        hashTable[hashIndex].count = 1;
        hashTable[hashIndex].head = newNode; // head를 교체
    }
    else {
        while (last->link != NULL) last = last->link;// 현재 리스트의 마지막 노드를 찾음
        last->link = newNode;                        // 새 노드를 마지막 노드(temp)의 다음 노드로 연결
        hashTable[hashIndex].count++;
    }
}

// 버켓에서 슬롯노드 x를 탐색하는 연산
bucketNode* searchSlot(bucket* hashTable, int BUCKET_SIZE, int x) {
    int hashIndex = hashFunction(BUCKET_SIZE, x);
    bucketNode* temp;
    temp = hashTable[hashIndex].head;
    while (temp != NULL) {
        if ((temp->key) == x) return temp;
        else temp = temp->link;
    }
    return temp;
}

// 체이닝 구조의 해시테이블 출력
void printHashTableAll(bucket* hashTable, int BUCKET_SIZE) {
    int i,j;
    bucketNode* p;
    printf("₩n₩n============ HashTable ============= ₩n");
    for (i = 0; i < BUCKET_SIZE ; i++) {
        printf("Bucket[%d] : ", i);
        p = hashTable[i].head;
        while(p != NULL) {
              printf(" [ %d%c ] ", p->key, p->value);
              p = p->link;
        }  printf("₩n");
    }
    printf("=================================== ₩n₩n");
}

void printHashTable(bucket* hashTable, int BUCKET_SIZE, int key) {
    bucketNode* p;
    int hashIndex = hashFunction(BUCKET_SIZE, key);

    printf("₩n₩n======= HashTable [key: %d]======== ₩n", key);
    printf("Bucket[%d] : ", hashIndex);
    p = hashTable[hashIndex].head;
    while (p != NULL) {
```

```
        printf(" [ %d%c ] ", p->key, p->value);
        p = p->link;
    }
    printf("₩n================================== ₩n₩n");
}
```

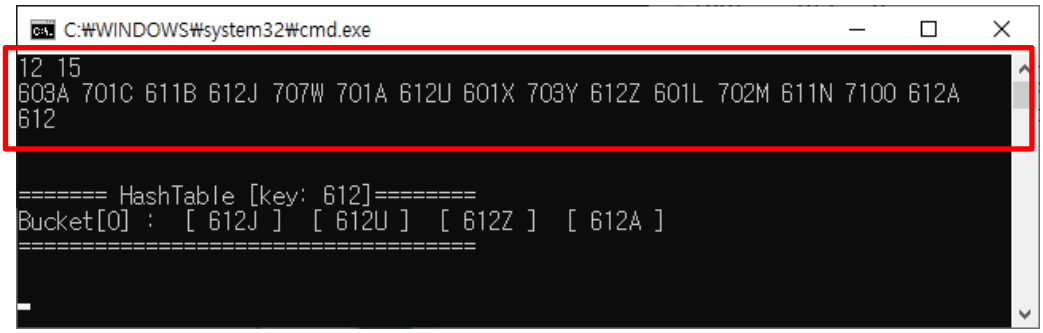
mailHashing.c

```
//응용예제17. 우편물 주소 해싱
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define MAX 20
#include "hash.h"
#define FL 2  //사무실 사용 층 개수

typedef struct {
        int key;
        char value;
}mailAddress;

int main(void) {
        int N, M, K, i,j;
        mailAddress mails[MAX];
        bucket* hashTable[2];
        bucket* hashTable600, *hashTable700;
        bucketNode* temp;

        //입력: 첫째 줄  => 사무실개수 N, 우편물 개수 M
        scanf("%d%d", &N, &M);
        //입력: 둘째 줄 => 우편물 주소 M개
        for (i = 0; i < M; i++)
                scanf("%d%c", &mails[i].key, &mails[i].value);
        //입력: 셋째 줄  => 우편물 배달할 사무실번호
        scanf("%d", &K);

        //해시 테이블 생성
        for(i=0; i<FL; i++)
                hashTable[i] = (bucket*)malloc(N * sizeof(bucket));

        //해시 테이블 초기화
        for(i=0; i<FL; i++)
                for (j = 0; j < N; j++) {
                        hashTable[i][j].count = 0;
                        hashTable[i][j].head = NULL;
                }
        //해시 테이블에 메일 저장
        for (i = 0; i < M; i++) {
                if ((mails[i].key/100) == 6) addSlot(hashTable[0], N, mails[i].key, mails[i].value);
                else addSlot(hashTable[1], N, mails[i].key, mails[i].value);
        }

        ////해시 테이블 내용 출력 : 전체
        //for (i = 0; i < FL; i++)
        //      printHashTableAll(hashTable[i], N);
```

```
        //해시 테이블 내용 출력 : 선택한 주소에 대한 버킷만 출력
        if ((K / 100) == 6) i = 0;
        else i = 2;
        temp = searchSlot(hashTable[i], N, K);
        printHashTable(hashTable[i], N, temp->key);

        getch(); return 0;
}
```

[실행화면]

입력



[실행화면2] 6층과 7층에 대해 구성한 전체 해시테이블 출력 (mailHashing.c에서 주석 해제하고 실행)

입력