**Student Number: 202135760**

**Student Name: 김현지**

**Problem 1**

* **Screenshots**

**텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

* **Code**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

void main() {

int scheduleNum; //일정 개수

//char startAt[100][6];//

//char endAt[100][6];

char scheduleTitle[100][256] = { "", };//일정 이름

int startAtSplit[100][2];//시작 시각 index: 0 = hour, 1 = minutes

int endAtSplit[100][2];//종료 시각 index: 0 = hour, 1 = minutes

//계산을 위해 분으로 포맷

int startFormatTime[100];

int endFormatTime[100];

int napTime;//낮잠시간

int maxNapTime = 0;//최대 낮잠 시간

int maxNapHour;//최대 낮잠 시간을 hour, minutes으로 분리

int maxNapMin;

int maxNapIndex = 0;

//스케줄 개수 입력받기

do {

printf("Enter your schedule number(<100) : \n");

scanf("%d", &scheduleNum);

} while (scheduleNum >= 100 || scheduleNum <= 0); //스케줄 개수가 조건에 맞을때만 다음으로 넘어가기

for (int i = 0; i < scheduleNum; i++) {//스케줄 입력받기

do {

printf("schedule #%d (All times will be greater than or equal 10:00 and less than or equal to 18:00:\n", i);

scanf("%d:%d %d:%d ", &startAtSplit[i][0], &startAtSplit[i][1], &endAtSplit[i][0], &endAtSplit[i][1]);

gets\_s(scheduleTitle[i]);

//계산을 위해 분으로 포맷

startFormatTime[i] = startAtSplit[i][0] \* 60 + startAtSplit[i][1];

endFormatTime[i] = endAtSplit[i][0] \* 60 + endAtSplit[i][1];

} while (startFormatTime[i] < 600 || (endFormatTime[i] > 1800));//입력 시간이 10:00 ~ 18:00

}

for (int i = 1; i < scheduleNum; i++) {//최대 낮잠 시간&그때의 index 구하기

napTime = startFormatTime[i] - endFormatTime[i - 1];

if (maxNapTime < napTime)

{

maxNapTime = napTime;

maxNapIndex = i;

}

}

maxNapHour = maxNapTime / 60;

maxNapMin = maxNapTime % 60;

if(maxNapTime >= 60)//maxNapTime이 1시간 넘어갈때

printf("the longest nap starts at %02d:%02d and will last for %d hours and %d minutes", endAtSplit[maxNapIndex - 1][0], endAtSplit[maxNapIndex - 1][1], maxNapHour, maxNapMin);

else

printf("the longest nap starts at %02d:%02d and will last for %d minutes", endAtSplit[maxNapIndex - 1][0], endAtSplit[maxNapIndex - 1][1], maxNapMin);

}

**Problem 2**

* **Screenshots**

**텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명**

* **Code**
* #include <stdio.h>
* #include <stdlib.h>
* #include <string.h>
* int pancakeCnt; //입력값 개수
* int flipIndex[60]; //flip한 위치 index를 순서대로 저장
* int flipCnt = 0;//flip 횟수
* int\* pancake(int\*, int);//팬케이크 알고리즘
* int\* flip(int[], int);//flip
* int findMax(int[], int);//max를 찾아서 그 index를 반환
* void printData(int[]);//전체 프린트
* void main() {
* int data[30];//입력값
* int\* result;//결과값
* int temp;//입력값 임시 저장
* int i = 0;
* //입력
* printf("enter pancake(end: -1):");
* for (i = 0; i < 30; i++) {
* scanf\_s("%d", &temp);
* if (temp == -1)
* break;
* else
* data[i] = temp;
* }
* pancakeCnt = i;
* result=pancake(data, i-1);
* //원래 값
* printData(data);
* //결과값
* printf("( ");
* printData(result);
* printf(")\n");
* //flip위치
* int j = -1;
* do{
* printf("%d ", flipIndex[++j]);
* } while (flipIndex[j] != 0);
* }
* int\* pancake(int\* data, int currentIndex) {//팬케이크 알고리즘: data = 입력값, currentIndex = 현재 마지막 위치
* if (currentIndex <= 0)//종료 조건
* return data;
* int maxIndex= findMax(data, currentIndex);//0-currentIndex까지 max값 찾기
* if (maxIndex != currentIndex) {//max가 이미 젤 뒤에 있으면 한 턴 넘기기
* if (maxIndex != 0) {//max가 처음에 위치한다면 첫번째 flip 넘기기
* data = flip(data, maxIndex);//0-maxIndex까지 flip => max를 처음으로 이동
* flipIndex[flipCnt++] = pancakeCnt - maxIndex;
* //printf("============flip1=============\n");
* //printData(data);
* }
* data = flip(data, currentIndex);//0-currentIndex까지 flip => max를 마지막으로 이동
* flipIndex[flipCnt++] = pancakeCnt - currentIndex;
* //printf("============flip2=============\n");
* //printData(data);
* }
* else
* currentIndex = currentIndex - 1;
* pancake(data, currentIndex - 1); //currentIndex-1: 이미 정렬된 max제외하고 다시 반복
* }
* int findMax(int data[], int currentIndex) {
* int max =0;
* int maxIndex = 0;
* for (int i = 0; i <= currentIndex; i++) {
* if (max < data[i]) {
* max = data[i];
* maxIndex = i;
* }
* }
* return maxIndex;
* }
* int\* flip(int data[], int currentIndex) {
* int\* filpData = (int\*)malloc(sizeof(int) \* pancakeCnt);//동적할당
* memset(filpData, 0, currentIndex \* sizeof(int));//초기화
* for (int i = 0; i <= currentIndex; i++) {//flip
* filpData[i] = data[currentIndex - i];
* }
* for (int j = currentIndex + 1; j < pancakeCnt; j++) {//이미 정렬된 부분 그대로 적어주기
* filpData[j] = data[j];
* }
* return filpData;
* free(filpData);
* }
* void printData(int data[]) {
* for (int i = 0; i < pancakeCnt; i++) {
* printf("%d ", data[i]);
* }
* //printf("\n");
* }