4190.103A-001: Programming Practice Lab. 11

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

Spring 2023

Lab. 10 실습 풀이

실습 1

Q. 2의 보수

어떤 수의 2의 보수는 해당하는 숫자의 모든 비트를 반전시킨 뒤, 1을 더해 만들 수 있다.
이때, 32비트 기준으로 처음 표현했던 수와 그 2의 보수의 서로 다른 비트 수를 출력하라.

입력

- 첫째줄에 N이 주어진다.

출력

- N과 N의 보수의 서로 다른 비트 수를 출력한다.

```c
#include<stdio.h>

int main(){
    int N, count=0;
    int bit[32] = {0,};
    int bitReversed[32] = {0,};
    scanf("%d", &N);

    /*
        Step 1) N을 이진수로 바꾸어 bit[] 배열에 저장.
    */
    for (int i = 31; i >= 0; i--) {
        bit[i] = N / (1 << i);
        N = N % (1 << i);
    }

    /*
        Step 2) bit[] 배열과 1과 0을 반대로 bitReversed[] 배열에 저장한 후, 1을 더함.
                이 때 2가 된 경우 올림 연산 수행, (1인 경우 0으로, 0인 경우 1로)
    */
    bitReversed[0] += 1;
    for (int i = 0; i < 32; i++) {
        bitReversed[i] += !bit[i];
        if (bitReversed[i] > 1 && i != 31)
            bitReversed[i + 1]++;
        bitReversed[i] %= 2;
    }

    /*
        Step 3) bit[] 배열과 bitReversed[] 배열에서 서로 다른 원소의 갯수를 카운트.
    */
    for (int i = 31; i >= 0; i--) {
        if (bit[i] != bitReversed[i]) {
            count++;
        }
    }
    printf("%d",count);
    return 0;
}
```

실습 2

Q. 2의 제곱

자연수 N이 주어졌을 때, 2의 제곱수이면 1을 아니면 0을 출력해보자.
단, 제곱수인지 판별할 때 주어진 `mask` 변수를 반드시 활용한다.
(모듈러 연산(%) 사용 불가)

입력

- 첫째줄에 N이 주어진다.
- N은 최대 2^32 - 1 이다.

출력

- 1또는 0을 출력한다.

```c
#include<stdio.h>
#define mask 0x01

int isPowerOfTwo(unsigned int N){
    // N의 최댓값이 2^32 -1 이라는 것과
    // shift 연산을 활용하자.
    int count = 0;
    for (int i = 0; i < 31; i++) {
        if (N & (mask << i))
            count++;
    }
    if (count == 1) return 1;
    return 0;
}
int main(){
    unsigned int N=0;
    scanf("%u", &N);
    printf("%d",isPowerOfTwo(N));
    return 0;
}
```

실습 3

Q.두 시간대의 차이

두 개의 시간을 입력받고, 그 차이를 출력해보자.

입력

- `시간 : 분 : 초` 의 형태로 첫째 줄과 둘째 줄에 두 개의 시간이 입력된다.
- 시간은 24시간제를 기준으로 한다.
- 입력되는 두 시간은 다음날로 넘어가지 않으며, 더 나중의 시간대가 첫째줄에 입력될 수도 있고 둘째줄에 입력될 수도 있다.

출력

- `시간 : 분 : 초` 의 형태로 그 차이를 출력한다.
- 분 과 초의 경우 10 이하일 경우 두 자리로 출력한다. (`%02d` 사용)

```c
void diffBetweenTimes(int *start,int *stop,int *diff){
    for (int i = 2; i >= 0; i--) {
        if (start[i] > stop[i]) {
            diff[i] += stop[i] + 60 - start[i];
            if (i != 0) diff[i - 1]--;
        } else {
            diff[i] += stop[i] - start[i];
        }
    }
}

int main(){
    int start[3], stop[3];
    int diff[3] = {0, };
    char ch1, ch2;

    /*
        diffBetweenTimes 함수에서 차이를 계산해 diff에 저장합니다.
    */
    scanf("%d %c %d %c %d", &start[0], &ch1, &start[1], &ch2, &start[2]);
    scanf("%d %c %d %c %d", &stop[0], &ch1, &stop[1], &ch2, &stop[2]);
    if (start[0] < stop[0])
        diffBetweenTimes(start, stop, diff);
    else if (stop[0] < start[0])
        diffBetweenTimes(stop, start, diff);
    else if (start[1] < stop[1])
        diffBetweenTimes(start, stop, diff);
    else if (stop[1] < start[1])
        diffBetweenTimes(stop, start, diff);
    else if (start[2] < stop[2])
        diffBetweenTimes(start, stop, diff);
    else if (stop[2] < start[2])
        diffBetweenTimes(stop, start, diff);

    // Result
    printf("%d : %02d : %02d\n",diff[0],diff[1],diff[2]);
    return 0;
}
```

실습 4

어떤 마을에 N명의 사람들이 살고 있다.
각 사람들은 이마에 자연수가 적혀있는데, 두 사람에 대해서 이마의 자연수를 이진수로 바꾼 후 각 자리 수가 같으면 0, 다르면 1로 표시해
그 값을 다시 10진수로 바꾼 값이 서로에 대한 신뢰도이다.
마을 사람들끼리의 신뢰도 합을 구해보자.

```
1 0 0 1 1 = 19
0 1 0 1 0 = 10
--------------
1 1 0 0 1 = 25
```

입력

- 첫째줄에 거주민의 수 N이 주어진다.
- 다음 N개의 줄에 자연수가 입력된다.
- N은 1보다 크고 2^32 - 1 보다 작거나 같다.

출력

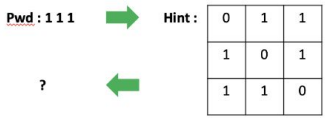
- 신뢰도 합을 출력한다.

```c
int solve(int N, int M) {
    return N ^ M;
    int B1[32];
    int B2[32];
    int count = 0;
    for (int i = 31; i >= 0; i--) {
        B1[i] = N / (1 << i);
        N = N % (1 << i);
    }
    for (int i = 31; i >= 0; i--) {
        B2[i] = M / (1 << i);
        M = M % (1 << i);
    }
    for (int i = 31; i >= 0; i--) {
        if (B1[i] != B2[i]) {
            count += (1 << i);
        }
    }
    return count;
}

int main(){
    int N;
    scanf("%d", &N);

    int INPUT[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &INPUT[i]);
    }
    int sum = 0;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            sum += solve(INPUT[i], INPUT[j]);
        }
    }
    printf("%d\n", sum);
```

Lab. 10 과제 풀이

과제 1

영수는 창고 비밀번호를 까먹어 오래전 힌트를 적어놓은 노트를 펼쳤다. 비밀 번호 길이는 N이고, 이에 대한 힌트로 N x N 크기의 2차원 행렬이 적혀 있었다. 시간이 오래 지나 행렬의 주 대각선은 모두 지워져 있었고, 나머지 숫자만 남아 있었다. 나머지 숫자를 사용해 원래 비밀번호를 찾아보자.



- 비밀번호는 음이 아닌 정수로 이루어져 있다.
- 행렬의 i번째 행 j열에 적힌 숫자는 비밀번호 i 번째 숫자와 j번째 숫자의 비트연산 and를 수행한 결과값이다.
- N은 1보다 크거나 같고, 1000보다 작은 정수이다.

입력

- 첫째줄에 N이 주어진다.
- 다음 N개의 줄에 행렬의 각 원소가 N개씩 주어진다.
- 지워진 숫자인 주대각선은 0으로 입력된다.

출력

- 0번째 자리부터 N-1번째 자리까지의 비밀번호를 띄어쓰기로 구분하여 한 줄로 출력한다.
- 답은 항상 존재하도록 주어지며, 여러개 존재하는 경우 해당 자리의 값이 최소가 되는 경우로 출력한다.

```c
int main(){
    int N;
    int ** hint;
    int * pwd;
    /*
        i 번째 행 j 번째 열의 숫자가 계산된 방법을 역으로 활용하여 비밀번호의 각 자리를 구해봅시다.
    */
    scanf("%d", &N);
    hint = (int **)malloc(sizeof(int *) * N);
    for (int i = 0; i < N; i++)
        hint[i] = (int *)malloc(sizeof(int) * N);
    pwd = (int *)malloc(sizeof(int) * N);

    for (int i = 0; i < N; i++) {
        pwd[i] = 0;
        for (int j = 0; j < N; j++) {
            scanf("%d", &hint[i][j]);
        }
    }

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            pwd[i] = pwd[i] | hint[i][j];
        }
    }

    for (int i = 0; i < N; i++)
        printf("%d ", pwd[i]);

    return 0;
}
```

과제 2

Q. N개의 보트

N개의 보트가 목적지를 향해 달리고 있다.
각 보트는 20개의 일렬로 된 자석이 있고, 한 개의 좌석에는 한 명의 사람만 탈 수 있다.
모든 보트들은 여러 섬의 정착지에 들러 사람을 태우거나 하차시킨다.
보트의 번호를 1번부터 N번으로 매길 때(좌석 번호 또한 1부터 N), 어떤 보트에 대해 M개의 명령이 주어진다.
명령어는 4종류로(`1, 2, 3, 4`), 각 번호가 의미하는 바는 아래와 같다.

<명령어>

`1 a b` : a번째 보트에 b번째 좌석에 사람을 태운다. 이미 사람이 있다면 명령어를 무시한다.

`2 a b` : a번째 보트에 b번째 좌석에 앉은 사람이 하차한다. 이미 사람이 없다면 명령어를 무시한다.

`3 a` : a번째 보트에 앉아있는 승객들이 모두 한 칸씩 뒤로 간다. 20번째 승객이 있었다면 그 사람은 하차시킨다.

`4 a` : a번째 보트에 앉아있는 승객들이 모두 한 칸씩 앞으로 간다. 1번째 승객이 있었다면 그 사람은 하차시킨다.

```c
int boat[100001] = {0,};

int countBoat(int N){
    // 중복되는 수를 제외하고 그 숫자를 카운트
    int count = 0;
    for (int i = 0; i < N; i++) {
        int flag = 1;
        for (int j = 0; j < i; j++) {
            int compare = ((1 << 20) - 1);
            if ((boat[i] & compare) == (boat[j] & compare)) {
                flag = 0;
                break;
            }
        }
        if (flag)
            count++;
    }
    return count;
}

int order(int num,int a,int b){
    // 명령어 번호 num에 따라서 다른 수행
    switch(num) {
        case 1:
            boat[a] = boat[a] | (1 << b);
            break;
        case 2:
            boat[a] = boat[a] & ~(1 << b);
            break;
        case 3:
            boat[a] = boat[a] << 1;
            break;
        case 4:
            boat[a] = boat[a] >> 1;
            break;
    }
}
```

```c
int main(){
    int N,M;
    scanf("%d %d", &N, &M);

    /*
        step 1 ) M번의 명령을 받아 번호에 따라 맞게 수행 해줍니다.
                 이 때 shift 연산(<<, >>)과 and 연산(&) or 연산(|) not 연산(~)을 활용할 수 있습니다.
    */
    for (int i = 0; i < M; i++) {
        int j, k, l;
        scanf("%d %d", &j, &k);
        if (j == 1 || j == 2)
            scanf("%d", &l);
        order(j, k, l);
    }
    printf("%d", countBoat(N));

    /*
        step 2 ) 보트에 타고 있는 승객 패턴에서 중복되는 패턴을 제외하고 그 수를 세어 줍니다.
                 boat[] 배열에 각 원소는 정수값이므로, 정렬을 한 후 세 주거나, 각 원소당 최대값을 계산해
                 check 배열을 만들어 세 주는 등, 다양한 방법이 있습니다.
    */
    return 0;
}
```

실습

실습 공통

- 구조체 변수에서 초기화해야 할 값이 있는지 확인한다.

실습 2

- 구조체 배열을 선언한다.
- 학생의 수와 학생에 대한 정보를 입력받고, 구조체 배열의 정보를 채워넣는다.
  - scanf("%d", &students[i].student_id);
- 구조체 배열의 구조체 변수들을 정렬한다.
  - 기존 정렬 방법을 그대로 사용하면 됨.

실습 4

```
typedef struct {
    char cabinet_id;
    int max_load;
    int taken;
} CABINET;

typedef struct {
    char student_id;
    int backpack_weight;
    CABINET * cb;
} STUDENT;
```

- CABINET 구조체 변수
  - cabinet_id (배열 상 인덱스)
  - max_load(최대 적재 가능 무게)
  - taken (사용 여부)
- STUDENT 구조체 변수
  - student_id (배열 상 인덱스)
  - backpack_weight (학생 가방의 무게)
  - cb (사용중인 CABINET을 가리키는 포인터)
- student_id가 적은 순으로 최대 적재 가능 무게와 가방의 무게 차가 적은 CABINET을 선택

실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 다음주 수요일 23:59 까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
- 제출 방법
  - Elice의 Submit 기능 활용