# Introduction to Computer and Lab

# Homework #8

**Due date: Jun 5, 2016**

**학번: 201404051**

**이름: 정 용 석**

## 1. 두 집합의 합집합 출력

### 1.1 Solution

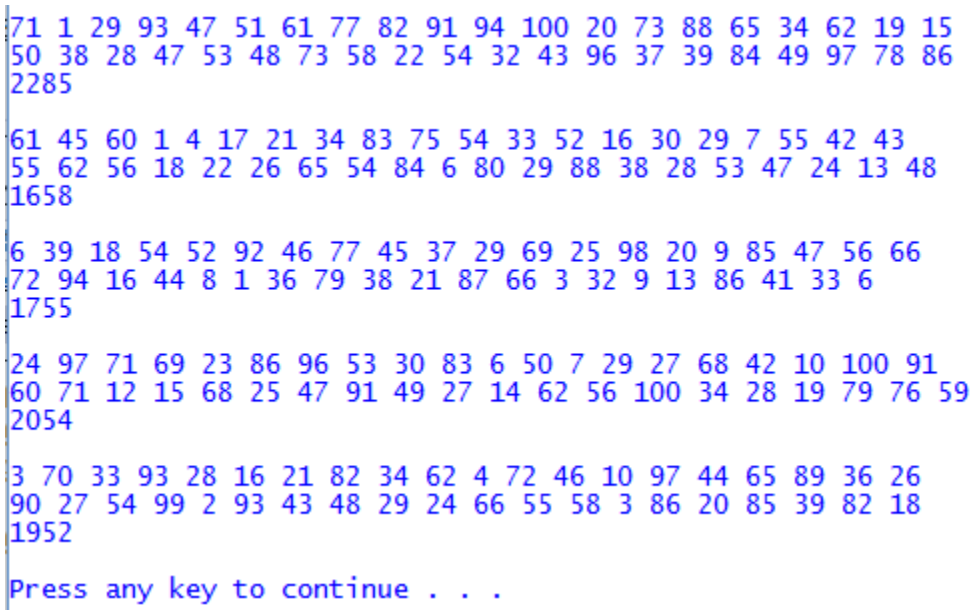
**크기 20 의 랜덤 수 집합 2 개를 생성하고, 앞에서부터 한 개씩 원소의 합을 더해가고, 이를 출력한다.**

### 1.2. Source code

```
void findUnionSet()    {
        //2개의 합집합의 합 출력
        int set1[SIZE];
        int set2[SIZE];
        int i, j, check;
        int sum = 0;
        //generateRandomSet함수와 동일
        for (i = 0; i < SIZE; i++)
        {
                while (1) {
                        set1[i] = rand() % 100 + 1;
                        set2[i] = rand() % 100 + 1;
                        check = 0;
                        for (j = 0; j < i; j++) {
                                if ((set1[j] == set1[i]) || (set2[j] == set2[i])) {
                                        check = 1;
                                        break;
                                }
                        }
                        if (!check)
                                break;
                }
        }
        //배열의 첫번째 원소들로부터 한개 씩 더한다.
        for (i = 0; i < SIZE; i++)
                sum += set1[i] + set2[i];
        //출력
        for (i = 0; i < SIZE; i++)
                        printf("%d ", set1[i]);
        printf("₩n");
```

```
        for (i = 0; i < SIZE; i++)
                printf("%d ", set2[i]);
        printf("₩n");
        printf("%d₩n", sum);
}
```

### 1.3. Result (snapshot)




## 2 Histogram 출력

### 2.1. Solution

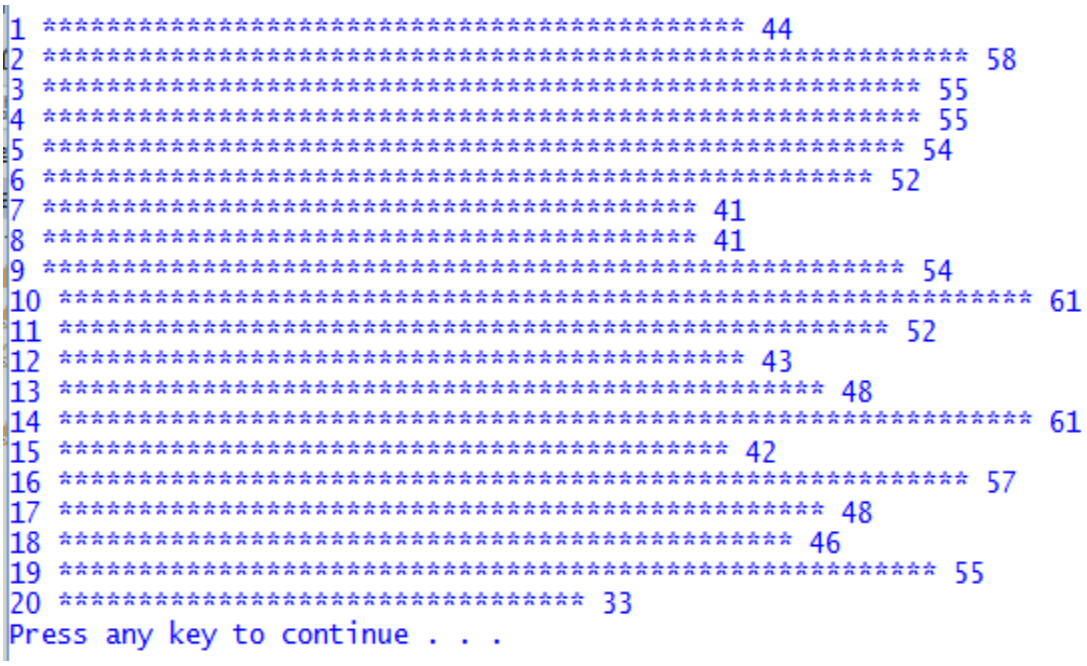
**크기가 1000 인 배열 안에 1 부터 20 사이의 값을 가지는 난수를 생성하고, 이중 반복 문을 통하여 각 배열의 index+1 값이 각각 몇 번 나오는지 확인하고 크기 20 의 배열에 각각 빈도 수를 저장한다. 각각의 빈도 수만큼 반복 문을 돌려서 '*'를 출력하면 된다.**

### 2.2. Source code

```
void printHistogram(){
        //Histogram 출력
        int frequency = 0; //빈도수
        int freqSet[20];
        int randNumber[1000];
        int i, j;
        int sum = 0;
        //1000개의 난수 생성 후 배열에 삽입
        for (i = 0; i < 1000; i++)
                randNumber[i] = rand() % 20 + 1;
        //1부터 20까지 난수 배열과 비교하여 빈도 수 계싼
```

```
	for (i = 0; i < 20; i++)
	{
		for (j = 0; j < 1000; j ++)
			if (randNumber[j] == i + 1) {
				frequency++;
			}
		//빈도 수 계산 후 빈도 수 배열에 삽입
		freqSet[i] = frequency;
		frequency = 0;
	}
	//각 숫자별 빈도 수 *로 출력
	for (i = 0; i < 20; i++) {
		printf("%d ", i + 1);
		for (j = 0; j < freqSet[i]; j++) {
			printf("*");
		}
		printf(" %dWn", freqSet[i]);
	}
}
```

### 2.3. Result (snapshot)




## 3 배열을 이용한 성적표 출력

### 3.1. Solution

**학생 수가 5 명이기 때문에 크기가 5 인 배열과, 각 학생 당 과목을 저장 할 수 있도록 4 X 5 의 2 차원 배열을 만들고 사용자로부터 각각 입력을 받는다. 그리고 각 ID 와 과목 출력과 동시에 총점과 평균을 계산하여 함께 출력한다.**

### 3.2. Source code

```
void printScoreReport(){
	//배열을 이용한 성적표 출력
	int sum = 0, average; //총합, 평균
	int ID[5]; //ID 5개
	int score[4][5]; //과목 4개 X 학생 수 5명
	int i, j;

	//배열의 첫 행에 ID 값을 저장
	//배열의 각 열에 각 과목에 대한 점수 저장
	for (i = 0; i < 5; i++) {
		printf("Input ID: ");
		scanf("%d", &ID[i]);

		printf("Input Scores: ");
		scanf("%d %d %d %d", &score[0][i], &score[1][i], &score[2][i], &score[3][i]);
	}

	//ID와 ID별 과목 점수 출력과 동시에 총합과 평균 계산 및 출력
	printf("IDWtKORWtENGWtMATHWtC++WtTOTALWtAVGWn");
	for (i = 0; i < 5; i++) {
		printf("%dWt", ID[i]);
		for (j = 0; j < 4; j++) {
			printf("%dWt", score[j][i]);
			sum += score[j][i];
		}
		average = sum / 4;
		printf("%dWt%dWn", sum, average);
		sum = 0;
	}
}
```

### 3.3. Result (snapshot)

```
Input ID: 0
Input Scores: 0 1 2 3
Input ID: 1
Input Scores: 3 6 5 8
Input ID: 2
Input Scores: 6 9 7 4
Input ID: 3
Input Scores: 4 5 8 7
Input ID: 4
Input Scores: 6 9 8 5
ID      KOR     ENG     MATH    C++     TOTAL   AVG
0       0       1       2       3       6       1.50
1       3       6       5       8       22      5.50
2       6       9       7       4       26      6.50
3       4       5       8       7       24      6.00
4       6       9       8       5       28      7.00

Input ID: 89
Input Scores: 79 46 13 79
Input ID: 56
Input Scores: 23 56 89 79
Input ID: 32
Input Scores: 31 64 97 79
Input ID: 25
Input Scores: 25 14 36 29
Input ID: 1
Input Scores: 75 89 94 61
ID      KOR     ENG     MATH    C++     TOTAL   AVG
89      79      46      13      79      217     54.25
56      23      56      89      79      247     61.75
32      31      64      97      79      271     67.75
25      25      14      36      29      104     26.00
1       75      89      94      61      319     79.75

Input ID: 10
Input Scores: 20 30 40 50
Input ID: 20
Input Scores: 30 40 50 60
Input ID: 30
Input Scores: 40 50 60 70
Input ID: 40
Input Scores: 50 60 70 80
Input ID: 50
Input Scores: 60 70 80 90
ID      KOR     ENG     MATH    C++     TOTAL   AVG
10      20      30      40      50      140     35.00
20      30      40      50      60      180     45.00
30      40      50      60      70      220     55.00
40      50      60      70      80      260     65.00
50      60      70      80      90      300     75.00
```

```
Input ID: 10
Input Scores: 10 20 30 40
Input ID: 1
Input Scores: 30 20 65 98
Input ID: 3
Input Scores: 46 89 97 9
Input ID: 4
Input Scores: 96 85 74 89
Input ID: 89
Input Scores: 70 80 90 100
ID      KOR     ENG     MATH    C++     TOTAL   AVG
10      10      20      30      40      100     25.00
1       30      20      65      98      213     53.25
3       46      89      97      9       241     60.25
4       96      85      74      89      344     86.00
89      70      80      90      100     340     85.00

Input ID: 1
Input Scores: 99 88 77 44
Input ID: 2
Input Scores: 99 98 97 95
Input ID: 3
Input Scores: 100 100 100 1
Input ID: 4
Input Scores: 30 20 60 100
Input ID: 5
Input Scores: 12 45 78 89
ID      KOR     ENG     MATH    C++     TOTAL   AVG
1       99      88      77      44      308     77.00
2       99      98      97      95      389     97.25
3       100     100     100     1       301     75.25
4       30      20      60      100     210     52.50
5       12      45      78      89      224     56.00
```

## 4 홀수 마방진 만들기

### 4.1. Solution

마방진의 규칙과 예외 상황 등을 잘 정리하여 코드화 시키면 된다. 보통은 1 차원 배열에서 현재 인덱스 + 1 그리고 밑으로 한 칸 내려가야 하기 때문에 r 로 변의 길이만큼 +를 한다. 하지만 첫 번째로 생각해야할 예외가 그 자리에 숫자가 있을 때이다. 따라서 배열을 애초에 0 으로 초기화 하고 넣고자 하는 자리가 0 이 아닐 경우 그 위칸, 즉 인덱스 – 가로 변의 길이를 한다. 만약 현재 인덱스가 오른쪽 변의 끝에 있을 경우 인덱스 + 1 만 하면 되고, 만약 배열의 끝부분이면 가장 첫 부분에 넣기만 하면 된다.

### 4.2. Source code

```
void mabangjin()
{
	//마방진 출력
	int jin[10 * 10] = { 0 }; //최대 넓이 100
	int N, i;
```

```
	int index;

	scanf("%d", &N);
	//입력 값이 짝수이면 N+1
	if (N % 2 == 0)
		N = N + 1;
	//입력 값이 범위를 넘어가면 오류 문 출력
	while (N > 10){
		printf("N should be a number between 1 to 9₩n");
		printf("Try Again₩n");
		scanf("%d", &N);
	}

	//첫 value인 1의 index 계산
	index = N*N - (N / 2) - 1;

	for (i = 1; i <= N*N; i++) {
		jin[index] = i;
		//다음 index 값 계산
		/* 1.index+1의 값이 boundary일 시,
			a) 배열의 첫번째 자리가 비어있을 때,
				index는 0이 된다.
			b) 배열의 첫번째 자리가 안비어있으면
				index는 그림 상 그 윗칸(index - N)
			c) 둘 다 해당사항 없으면 index + 1;
		   2. 다음 index값이 비어있는 공간이 아니면 index-N
		   3. default는 오른쪽 한칸 이동 후 밑으로 한 칸
		*/
		if ((index + 1) % N == 0){
			if ((index + 1 == N*N)) {
				if (jin[0] == 0)
					index = 0;
				else
					index = index - N;
			}
			else
				index = index + 1;
		}
		else if (jin[((index + 1) + N) % (N*N)] != 0)
			index = index - N;
		else
			index = ((index + 1) + N) % (N*N);
	}
	//출력
	for (i = 0; i < N*N; i++) {
		if (i%N == 0 && i != 0)
			printf("₩n");
		printf("%d ", jin[i]);
	}
	printf("₩n");
}
```

### 4.3. Result (snapshot)

```
3
4 9 2
3 5 7
8 1 6

4
11 18 25 2 9
10 12 19 21 3
4 6 13 20 22
23 5 7 14 16
17 24 1 8 15

5
11 18 25 2 9
10 12 19 21 3
4 6 13 20 22
23 5 7 14 16
17 24 1 8 15

6
22 31 40 49 2 11 20
21 23 32 41 43 3 12
13 15 24 33 42 44 4
5 14 16 25 34 36 45
46 6 8 17 26 35 37
38 47 7 9 18 27 29
30 39 48 1 10 19 28

7
22 31 40 49 2 11 20
21 23 32 41 43 3 12
13 15 24 33 42 44 4
5 14 16 25 34 36 45
46 6 8 17 26 35 37
38 47 7 9 18 27 29
30 39 48 1 10 19 28

Press any key to continue . . .
```

## 5 같은 숫자 찾기

### 5.1. Solution

**같은 수가 2 번씩 나오기 때문에 배열의 크기/2 만큼의 수 까지만 쓸 수 있다. 배열을 0 으로 초기화 하고 1 부터 수를 넣을 때 2 개의 랜덤 인덱스를 계산하여 수를 삽입한다. 그리고 알맞게 출력한다.**

### 5.2. Source code

```
void SameNumberGame()
{
	int set[10 * 10] = { 0 }; //100넓이로 초기화(최대)
	int width, height;
	int area;
```

```
	int index, i;

	printf("Input width, height: ");
	scanf("%d %d", &width, &height);
	area = width * height;
	//넓이가 홀수 이면 오류 문 출력
	if (area % 2 != 0)
	{
		while (1) {
			printf("Area of the table should be even. Try Again₩n");
			printf("Input width, height: ");
			scanf("%d %d", &width, &height);
			area = width * height;
			if (area % 2 == 0)
				break;
		}
	}

	//1~전체 배열 원소 개수/2 까지 random index 2곳에 저장
	for (i = 1; i <= area/2; i++) {
		index = rand() % area;
		if (set[index] != 0) {
			//index값이 비어있지 않으면 빌 때까지 난수 생성
			while (set[index] != 0)
				index = rand() % area;
		}
		set[index] = i;
		index = rand() % area;
		if (set[index] != 0) {
			//index값이 비어있지 않으면 빌 때까지 난수 생성
			while (set[index] != 0)
				index = rand() % area;
		}
		set[index] = i;
	}

	//출력
	for (i = 0; i < area; i++)
	{
		if (i % width == 0 && i != 0)
			printf("₩n");
		printf("%d ", set[i]);
	}
	printf("₩n");
}
```

### 5.3. Result (snapshot)

```
Input width, height: 3 4
6 4 5
4 2 1
3 5 3
6 2 1

Input width, height: 1 2
1
1

Input width, height: 4 5
10 8 6 6
7 10 3 2
8 4 9 3
4 7 5 2
1 9 1 5

Input width, height: 7 8
26 16 21 27 6 15 2
20 25 1 2 11 17 26
19 10 22 13 6 25 4
3 18 5 7 13 21 19
27 5 18 20 10 4 12
14 23 15 8 24 8 23
1 9 9 28 12 14 16
11 22 28 7 24 3 17

Input width, height: 8 9
24 8 9 34 19 31 21 22
2 1 13 28 7 9 14 11
35 20 12 27 10 26 17 29
28 4 10 33 30 23 5 21
3 14 32 19 4 1 26 36
6 16 36 24 33 12 5 16
2 6 15 32 22 17 30 27
20 13 31 11 35 18 7 8
29 3 34 15 25 23 25 18

Press any key to continue . . .
```