창의적 소프트웨어 설계 실습 문제 2주차

제출 기한
- 채점서버 제출 : (1차) 10월 1일 토 11:59pm 까지
(2차) 10월 8일 토 11:59pm 까지

2-1. 진법변환 (radix notation)

입력된 수를 주어진 진법으로 표시하는 프로그램 작성.

- 양의 정수를 command line으로 입력받아 두번째 이후의 숫자들을 첫번째 수의 진법으로 표현한다.
- 진법은 2 에서 36 까지 가능하며 0-9 a-z 를 사용하여 숫자를 표현한다.
- 계속 입력을 받아 표시하도록 하며, radix로 비정상 값이 입력되면 종료한다.
- 주어진 skeleton에서 RadixNotation 함수를 구현한다.
    - (코드의 correctness 가 맞으면 skeleton code 를 사용하지 않아도 상관없음)

파일명 : radix_notation (.cc / .cpp / .c)
입력 : 양의 정수인 숫자와 진법.
출력 : 화면에 변환된 문자열을 출력.

```
$ ./radix_notation 1 123
$ ./radix_notation 10 123
123
$ ./radix_notation 8 10 250
12
372
$ ./radix_notation 16 250
fa
$ ./radix_notation 2 250
11111010
$ ./radix_notation 32 4 10 200 255 65536
4
a
68
7v
2000
```

Code skeleton (라인 수를 절약하기 위해 컨벤션은 맞추지 않았음)

```
// radix_natation.cc

#include <stdio.h>
#include <iostream>
#include <string>
```

```
using namespace std;

// Implement this function.
string RadixNotation(unsigned int number, unsigned int radix);

int main(int argc, char** argv) {
  if (argc < 2) return 1;
  unsigned int radix;
  sscanf(argv[1], "%u", &radix);
  if (radix < 2 || radix > 36) return 1;
  for (int i = 2; i < argc; ++i) {
    unsigned int number;
    sscanf(argv[i], "%u", &number);
    cout << RadixNotation(number, radix) << endl;
  }
  return 0;
}
```

```
// radix_natation.c

#include <stdio.h>
char g_radix_notation[1024];

// Implement this function.
char *RadixNotation(unsigned int number, unsigned int radix);

int main(int argc, char** argv) {
  if (argc < 2) return 1;
  unsigned int radix;
  sscanf(argv[1], "%u", &radix);
  if (radix < 2 || radix > 36) return 1;
  for (int i = 2; i < argc; ++i) {
    unsigned int number;
    sscanf(argv[i], "%u", &number);
    printf("%s\n", RadixNotation(number, radix));
  }
  return 0;
}
```

*** C언어로 프로그래밍 하는 경우 이슈**

- 함수 내에서 지역 변수로 배열을 선언하면 함수를 벗어 나오면서 사라지므로, 배열은 동적 할당 또는 전역 변수를 사용한다.

2-2. 간단한 점 그리기 (draw points)

주어진 개수만큼의 2차원 점을 화면에 출력하는 프로그램 작성

- 입력으로는 각 점의 x,y 좌표(음이 아닌 정수)가 주어진다.
- 화면에 점이 없는 위치에는 '.'을 점이 있는 위치에는 '*'를 출력한다.
- 좌상단이 (0,0) 이며 꼭 필요한 부분만 출력한다. 즉, 오른쪽과 아래쪽 모서리에는 항상 점이 있어야 한다. 좌표는 오른쪽이나 아래쪽으로 갈수록 1씩 증가한다.
- 점의 좌표가 음수가 입력되면 종료한다.

파일명 : draw_points (draw_points.cc)
입력 : 각 점의 x,y 위치
출력 : 각 점의 위치를 나타내는 도형

```
$ ./draw_points
3 1
....
...*
1 2
....
...*
.*..
0 1
....
*..*
.*..
-1 -1
$
```

Code skeleton

```
#include <stdio.h>
#include <stdbool.h>
#define      MAX_X  1024
#define      MAX_Y  1024
/**
  * draws a point given parameters as "*" and fills others as "."
  * @param[in] x    x coordinate of a point
  * @param[in] y    y coordinate of a point
  * @return true if given coordinate can be presented at the scrren
  */
bool DrawPoint(int x, int y) {
    /* TODO: implement here. */
    return false;
}

void main() {
    int x;
    int y;
    int screen[MAX_Y][MAX_X];
```

```
    while (true) {
        /* input of 2 integers from users to represent a point */
        scanf("%d %d", &x, &y);
        if (!DrawPoint(x, y)) {
            break;
        }
    }
}
```

2-3. 소인수 분해 (Prime Number Factorization)

자연수를 입력하면 자연수를 소인수 분해하여 출력하는 프로그램 작성

- 입력은 command line으로부터 자연수를 하나 받는다.
- 가장 작은 소인수부터 차례대로 출력한다
- 곱셈 연산자는 x로 표현하며, 제곱 연산자는 ^로 표현한다. (e.g. 3^3 x 5^2)
  - (*가 아니라 알파벳 x로 표기해야함)
- 1은 소인수가 아니므로 포함하지 않는다.
- 소수가 입력됐을 시에는, 소수^1로 표현한다. (e.g. 19^1)
- 자연수가 아닌 값(문자열, 실수 등)이 입력되면 출력 없이 프로그램을 종료한다.

파일명 : prime_factorization (prime_factorization.cc)
입력 : 자연수
출력 : 소인수분해된 결과를 출력 (연산자간 띄어쓰기에 유의)

```
$ ./prime_factorization 5
5^1
$ ./prime_factorization 12
2^2 x 3^1
$ ./prime_factorization 36
2^2 x 3^2
$ ./prime_factorization 720
2^4 x 3^2 x 5^1
```

Code skeleton

```
#include <iostream>
#include <stdio.h>

using namespace std;

// Implement this function
string PrimeFactorization(unsigned int _number);
```

```
int main(int argc, char** argv)
{
  if(argc < 1) return 1;

  unsigned int number;
  sscanf(argv[1], "%u", &number);

  cout << PrimeFactorization(number) << endl;
  return 0;
}
--------------------------------------------------------------------------------

#include <stdio.h>

char g_prime_notation[1024];

// Implement this function
char *PrimeFactorization(unsigned int _number);

int main(int argc, char** argv)
{
  if(argc < 1) return 1;

  unsigned int number;
  sscanf(argv[1], "%u", &number);

  printf("%s\n", PrimeFactorization(number));
  return 0;
}
```