Python SelfTest

작성자	노성진

ver 0.9.1

2022. 04. 25

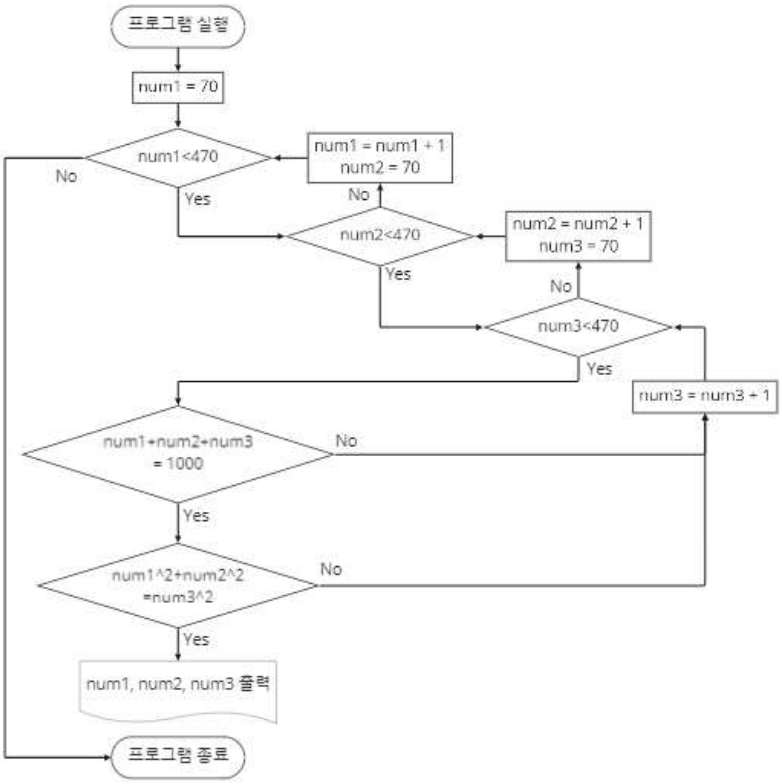
목차

<제목 차례>

1. 연산 프로그램

1) 피타고라스 조건 만족, 합 1000인 자연수 숫자 3개 찾기

(1) Flow Chart



(2) 소스코드

```
for num1 in range(70,470):
    for num2 in range(70,470):
        for num3 in range(70,470):
            if(num1 + num2 + num3 == 1000):  # 합이 1000일 때
                if(num1**2 + num2**2 == num3**2):  # 피타고라스 조건을 만족할 때
                    print("밑변 = {}, 높이 = {}, 빗변 = {} 이다.".format(num1,num2,num3))
# range(1,1000)으로 하면 연산이 너무 오래 걸려 원시 피타고라스 수를 활용해 70~470로 범위 제한
```

(3) 결과 화면

```
밑변 = 200, 높이 = 375, 빗변 = 425 이다.
밑변 = 375, 높이 = 200, 빗변 = 425 이다.
```

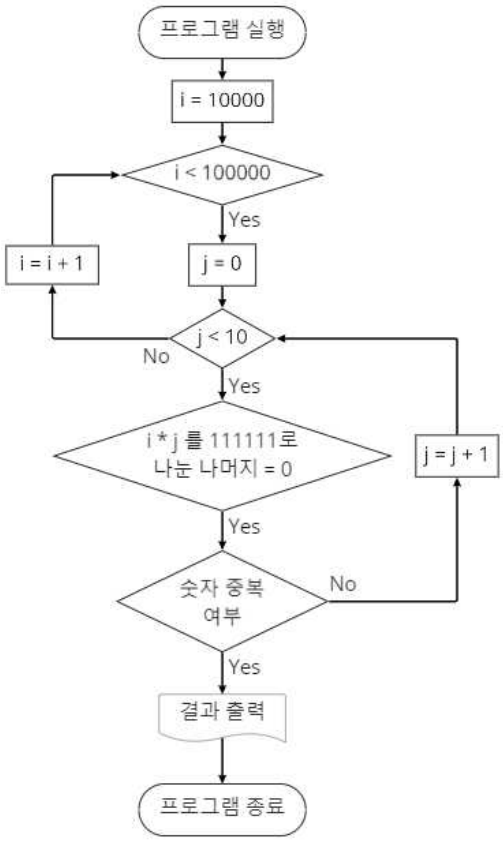
(4) 설명

70~470 범위 안에서 3개 숫자 합이 1000인 경우 중 피타고라스 조건을 만족하는 숫자를 출력합니다.

범위를 제한한 이유는 두 자리 이하의 원시 피타고라스 수 16쌍을 n배 했을 때 합이 1000이하인 값들의 최솟값이 70 최댓값이 470 이기 때문입니다.

2) ABCDE*F=GGGGGG인 ABCDEFG 구하기 (숫자 중복 없음)

(1) Flow Chart



(2) 소스코드

```
for i in range(10000,100000):
    for j in range(1,10):
        if((i * j) % 111111 == 0):
            check = str(i)+str(j)+str(i*j//111111)
            set_check = set(check)
            if len(set_check) == 7 :
                result = check
print("ABCDE = {}, F = {}, G = {}이다.₩nC+G는 {}이다".format(check[:5],check[5:6], check[6:], int(check[2:3])+int(check[6:])))
```

(3) 결과 화면

```
ABCDE = 95238, F = 7, G = 60이다.
C+G는 80이다
```

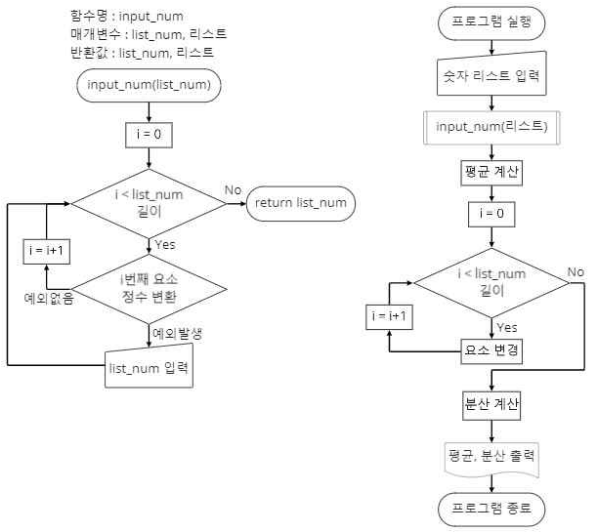
(4) 설명

반복문을 활용하여 조건을 만족하는 값을 구하는 프로그램입니다.

문제에 나오는 조건을 만족하는 숫자를 찾고 중복 제외 함수를 활용하여 중복 여부를 체크합니다. 길이가 변하지 않을 경우 즉, 중복된 값이 없을 경우 값을 출력합니다.

3) 평균과 분산 구하기

(1) Flow Chart



(2) 소스코드

```
def input_num(list_num) : # 리스트 요소가 숫자인지 확인 함수
    while True :
        try :
            for i in range(len(list_num)) : # 리스트 요소 정수 변환 후 반환
                list_num[i] = int(list_num[i])
            return list_num
        except : # 에러 발생시 재입력
            print("잘못 입력하셨습니다.")
            list_num = input("다시 입력해주십시오. : ").split()
list_num = input("숫자를 입력하시오.").split() # 숫자 입력
list_num = input_num(list_num) # 숫자 확인 함수 호출
result_avg = sum(list_num)/len(list_num) # 평균 계산
list_var = []
for i in range(len(list_num)) : # 분산 값을 구하기 위한 값 리스트 정리
    list_var.append((int(list_num[i]) - result_avg) ** 2)
result_var = sum(list_var)/len(list_var) # 분산 계산
print("{}의 평균은 {}이고 분산은 {}입니다.".format(list_num, result_avg, result_var))
```

(3) 결과 화면

```
숫자를 입력하시오.1 3 5 7
[1, 3, 5, 7]의 평균은 4.0이고 분산은 5.0입니다.
```

(4) 예외처리 화면

숫자를 입력하지 않은 경우

```
숫자를 입력하시오.1 3 5 !
잘못 입력하셨습니다.

다시 입력해주십시오. :
```

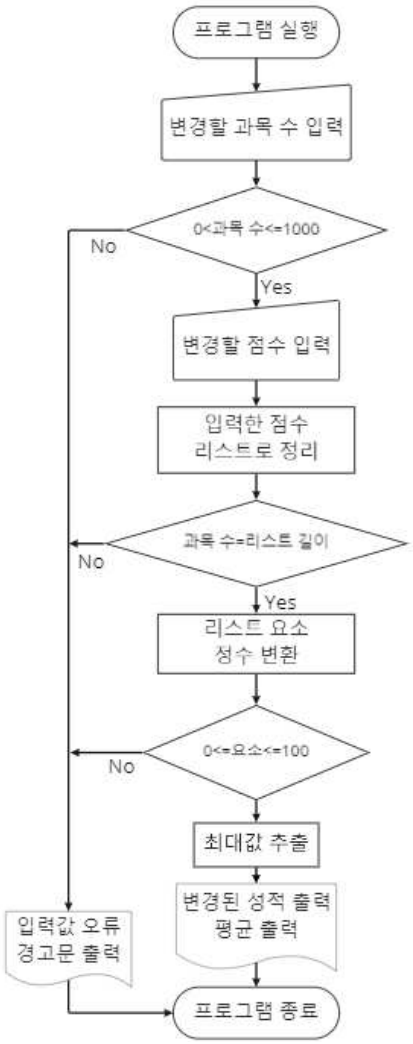
(5) 설명

평균과 분산을 구하는 프로그램입니다.

최소 2개 이상의 숫자를 입력하여 평균, 분산을 계산하고 숫자를 입력하지 않은 경우 다시 입력하는 input_num 함수를 만들어 예외처리 했습니다.

4) 과목 점수 변경하기

(1) Flow Chart



(2) 소스코드

```
subject = int(input("1000이하의 양의 정수를 입력하시오 > ")) # 과목 수 입력
input_pass = False  # 입력값 판별 기준
if(0 < subject <= 1000) :  # 과목 수 범위 제한
    score = list(input("100이하의 음이 아닌 정수를 {}개 입력하시오 > ".format(subject)).split())  # 점수 입력, 리스트 정리
    if (len(score) == subject):  # 과목 수에 맞게 입력했는지 확인
        for i in range(len(score)):  # 요소 정수 변환
            score[i] = int(score[i])
            if (0 <= score[i] <= 100):  # 점수 범위 확인
                input_pass = True
            else:
                print('잘못된 입력값입니다.100이하의 양의 정수를 입력해주십시오.')
                input_pass = False
                break
    else:
        print('입력한 과목 수와 다릅니다.')
else :
    print("잘못된 입력값입니다.1000이하의 양의 정수를 입력해주십시오.")
if (input_pass == True):
    max_score = max(score)  # 최댓값 추출
    for i in range(len(score)) :  # 점수 변경
        score[i] = score[i] / max_score * 100
    sum_score = sum(score)  # 총합 계산
    print('변경된 성적은 {}입니다.'.format(score))
    print("평균은 {}점 입니다.".format(sum_score / subject))
```

(3) 결과 화면

```
10000이하의 양의 정수를 입력하시오 > 3
100이하의 음이 아닌 정수를 3개 입력하시오 > 40 80 60
변경된 성적은 [50.0, 100.0, 75.0]입니다.
평균은 75.0점 입니다.
```

(4) 예외처리 화면

(4-1) 과목 수 입력에서 1000이하의 양의 정수를 입력하지 않은 경우

```
10000이하의 양의 정수를 입력하시오 > -1
잘못된 입력값입니다.10000이하의 양의 정수를 입력해주십시오.
```

(4-2) 점수 입력에서 100이하의 음이 아닌 정수를 입력하지 않은 경우

```
10000이하의 양의 정수를 입력하시오 > 3
100이하의 음이 아닌 정수를 3개 입력하시오 > -10 0 50
잘못된 입력값입니다.100이하의 양의 정수를 입력해주십시오.
```

(4-3) 과목 수와 점수 입력한 수가 틀릴 경우

```
10000이하의 양의 정수를 입력하시오 > 3
100이하의 음이 아닌 정수를 3개 입력하시오 > 10 50 60 80
입력한 과목 수와 다릅니다.
```

(5) 설명

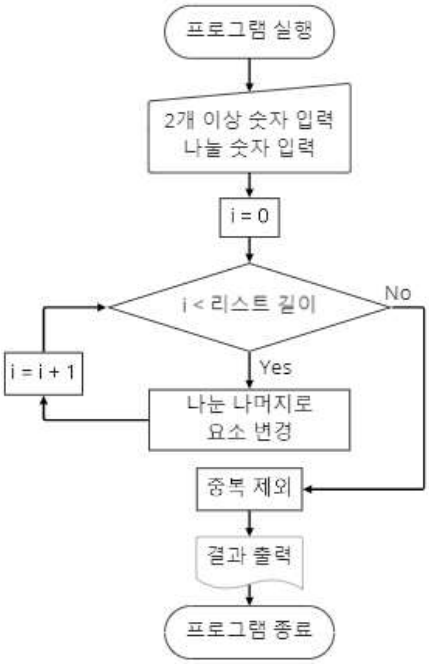
점수를 조건에 맞게 변경하는 프로그램입니다.

과목 수를 입력하고 각 과목 점수를 공백으로 구분하여 입력합니다. 입력한 점수 중 최댓값을 추출하여 (실제 점수/최댓값)*100 으로 모든 점수를 변경합니다.

과목 수, 점수 입력에서 범위를 벗어난 값을 입력하거나 입력한 과목 수와 점수 개수가 틀릴 경우 해당 오류를 출력하고 프로그램을 종료합니다.

5) 나눈 나머지가 중복될 경우 제외하기

(1) Flow Chart



(2) 소스코드

```
f_num = input("최소 2개 이상의 숫자를 입력해주십시오. : ").split() # 최소 2개 이상 숫자 입력
s_num = input("숫자를 입력해주십시오. : ") # 나눌 숫자 입력
f_len = len(f_num) # 처음 입력한 숫자 개수
for i in range(len(f_num)) : # 입력한 숫자로 나눈 나머지로 요소 변경
    f_num[i] = int(f_num[i]) % int(s_num)
set_num = list(set(f_num)) # 중복 제외
s_len = len(set_num) # 중복 제외한 리스트 길이
print('나머지 : {} 중복 제외된 나머지 : {}'.format(f_num, set_num))
print('나머지가 서로 다른 값은 {}개 중 {}개 입니다.'.format(f_len, f_len-s_len))
```

(3) 결과 화면

```
최소 2개 이상의 숫자를 입력해주십시오. : 10 11 12 13
숫자를 입력해주십시오. : 2
나머지 : [0, 1, 0, 1] 중복 제외된 나머지 : [0, 1]
나머지가 서로 다른 값은 4개 중 2개 입니다.
```

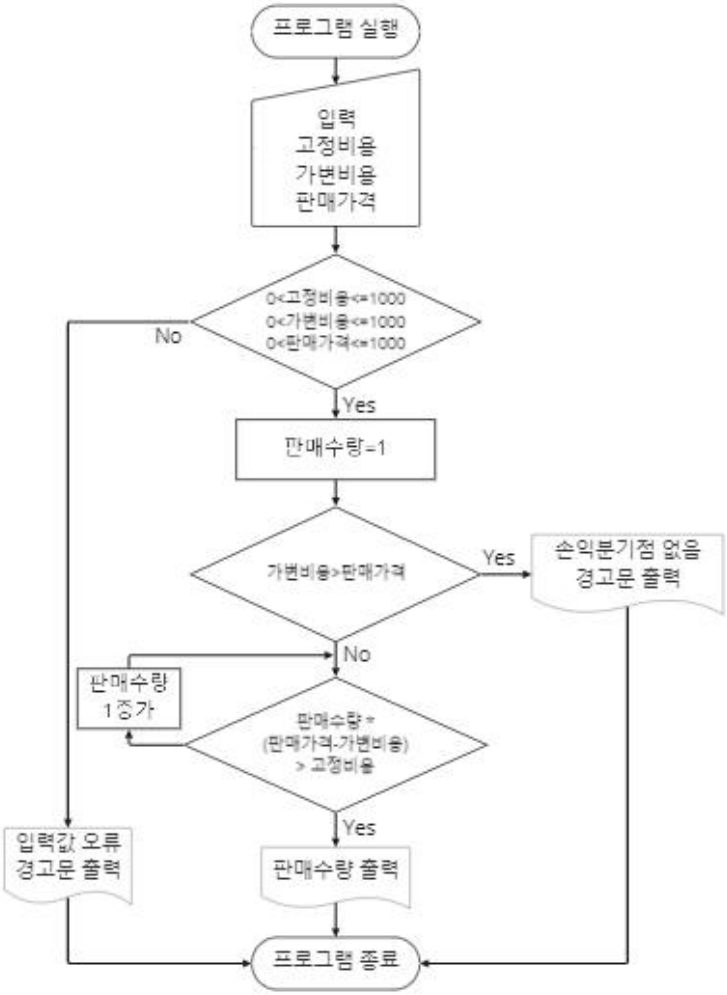
(4) 설명

나머지의 중복을 체크하여 제외하는 프로그램입니다.

최소 2개 이상의 숫자를 입력받고 나눌 숫자를 입력받아 나머지를 리스트로 정리합니다. 나머지 중복을 제외하여 이전 리스트 길이와 비교합니다.

6) 손익분기점 찾기

(1) Flow Chart



(2) 소스코드

```
fix_cost = int(input("고정비용을 입력하십시오.(10000이하 양의 정수)"))  # 고정비용 입력
variable_cost = int(input("가변비용을 입력하십시오.(10000이하의 양의 정수)"))  # 가변비용 입력
sell_cost = int(input("판매가격을 입력하십시오.(10000이하의 양의 정수)"))  # 판매가격 입력
sell_count = 1
stop = "0"
if(0 < fix_cost <= 1000 and 0 < variable_cost <= 1000 and 0 < sell_cost <= 1000 ): # 입력값이 10000이하 양의 정수일 경우
    if(variable_cost > sell_cost) : # 판매가격 설정 오류
        print("판매 가격이 가변비용보다 낮아 손익분기점이 없습니다.")
    else :
        while(True):
            if(fix_cost < (sell_cost - variable_cost) * sell_count) :  # 판매>비용인 지점
                print("손익분기점 판매 수량은 {}입니다.".format(sell_count))
                break
            else :
                sell_count += 1
else:
    print("잘못 입력하셨습니다. 10000이하의 양의 정수를 입력해주십시오.")
```

(3) 결과 화면

```
고정비용을 입력하십시오.(10000이하 양의 정수)1000
가변비용을 입력하십시오.(10000이하의 양의 정수)70
판매가격을 입력하십시오.(10000이하의 양의 정수)170
손익분기점 판매 수량은 11개입니다.
```

(4) 예외처리 화면

(4-1) 범위를 벗어난 값 입력한 경우

```
고정비용을 입력하십시오.(10000이하 양의 정수)1200
가변비용을 입력하십시오.(10000이하의 양의 정수)1500
판매가격을 입력하십시오.(10000이하의 양의 정수)170
잘못 입력하셨습니다. 10000이하의 양의 정수를 입력해주십시오.
```

(4-2) 판매가격이 가변비용보다 작은 경우

```
고정비용을 입력하십시오.(10000이하 양의 정수)1000
가변비용을 입력하십시오.(10000이하의 양의 정수)80
판매가격을 입력하십시오.(10000이하의 양의 정수)70
판매 가격이 가변비용보다 낮아 손익분기점이 없습니다.
```

(5) 설명

손익분기점을 찾는 프로그램입니다.

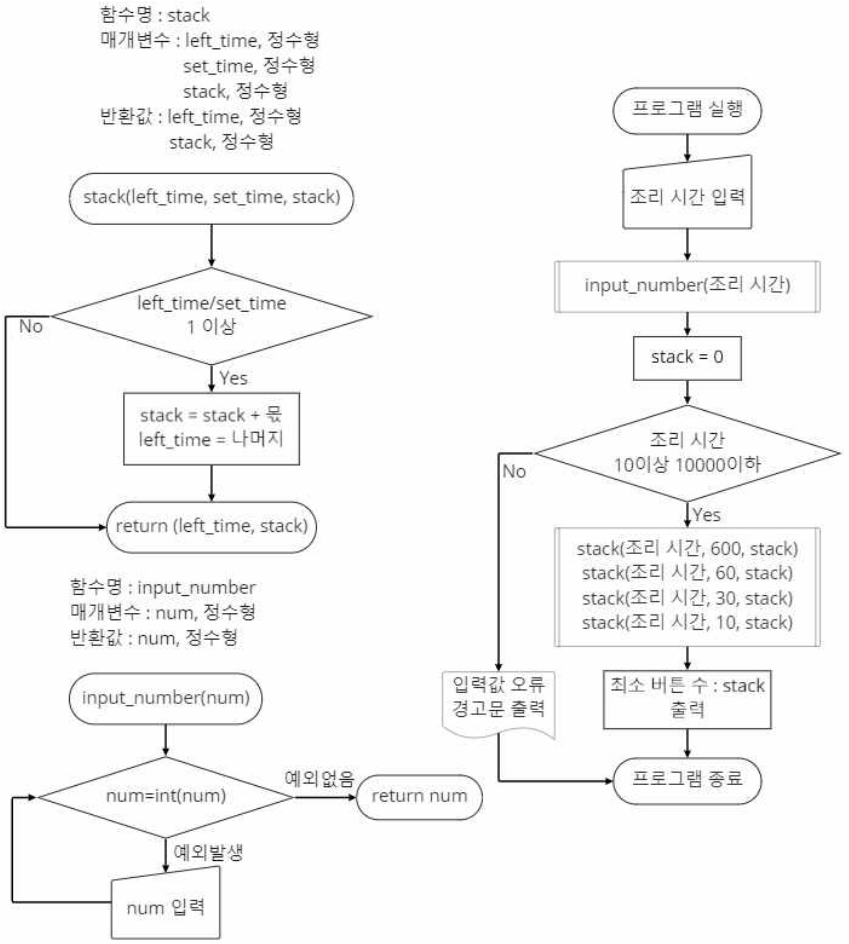
고정비용, 가변비용, 판매가격을 입력받고 손익분기점이 생기는 판매 수량을 계산하여 출력합니다.

각 입력 값이 제한 범위를 벗어난 경우 해당 오류를 출력하고 프로그램을 종료합니다.

손익분기점 계산식에서 가변비용이 판매가격보다 클 경우 손익분기점은 생길 수 없습니다. 이에 대한 오류 또한 경고문을 출력하고 프로그램을 종료합니다.

7) 전자레인지 최소 버튼 클릭으로 조리 시간 맞추기

(1) Flow Chart



(2) 소스코드

```python
def stack(left_time, set_time, stack) : # left_time을 set_time로 나눈 값의 몫, 나머지 반환 함수
    if(left_time // set_time >= 1) :
        stack += int(left_time // set_time) # 나눈 값의 몫 스택 추가
        left_time = left_time % set_time # 나눈 값의 나머지 반환
    return (left_time, stack)
def input_number(num) : # 입력값이 숫자인지 확인하는 함수
    while True :
        try :
            num = int(num) # 매개변수를 정수로 변환하고 반환
            return num
        except ValueError : # ValueError가 발생한 경우 다시 입력
            print("숫자를 입력해주십시오: ")
            num = input("다시 입력해주십시오: ")

second = input("몇 초를 조리하시겠습니까? : ") # 조리 시간 입력
second = input_number(second) # 숫자 확인 함수 호출
button_stack = 0
if(10 <= second <= 10000) :
    left_time = second
    (left_time,button_stack) = stack(left_time, 600, button_stack) # 600초로 나눠질 때 실행
    (left_time,button_stack) = stack(left_time, 60, button_stack) # 60초로 나눠질 때 실행
    (left_time,button_stack) = stack(left_time, 30, button_stack) # 30초로 나눠질 때 실행
    (left_time,button_stack) = stack(left_time, 10, button_stack) # 10초로 나눠질 때 실행
    print("{}초를 맞추기 위한 최소 버튼 수는 {}번 입니다.".format(second,button_stack))
else : # 범위 맞지 않는 값이 입력된 경우 오류 출력
    print("10에서 10000사이의 값을 입력해주십시오.")
```

(3) 결과 화면

```
몇 초를 조리하시겠습니까? : 100
100초를 맞추기 위한 최소 버튼 수는 3번 입니다.
```

(4) 예외처리 화면

범위에 맞지 않은 값을 입력한 경우

```
몇 초를 조리하시겠습니까? : 1
10에서 10000사이의 값을 입력해주십시오.
```

10초 단위가 아닌 값을 입력한 경우

```
몇 초를 조리하시겠습니까? : 101
10초 단위로 입력해주십시오.
```

(5) 설명

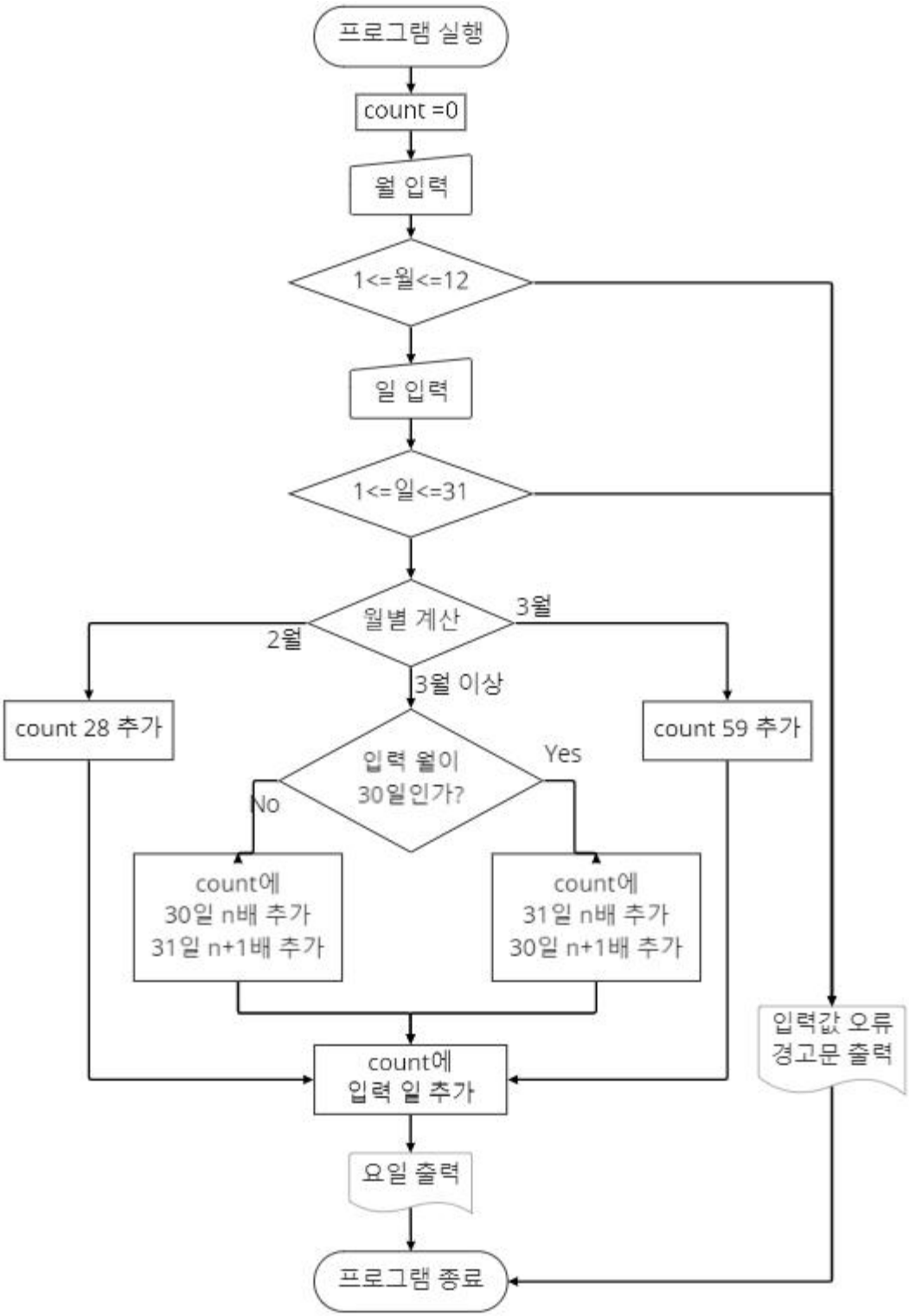
최소 버튼 클릭으로 조리 시간을 계산하여 클릭수를 출력하는 프로그램입니다.

버튼은 10초, 30초, 60초, 600초 4개로 구성되어 있으며 조리 시간은 10초~10000초 사이를 10초 단위로 입력할 수 있습니다. 입력된 조리 시간을 최소 버튼 클릭으로 계산하여 클릭수를 출력합니다.

조리 시간 입력 오류 발생시 경고문을 출력하고 프로그램을 종료합니다.

8) 특정 월, 일의 요일 확인하기

(1) Flow Chart



(2) 소스코드

기본 소스 코드

```
day_30 = [4, 6, 9, 11] # 30일인 월
day_31 = [1, 3, 5, 7, 8, 10, 12] # 31일인 월
day_of_week = ['월', '화', '수', '목', '금', '토', '일'] # 요일 리스트
count = 0
month = int(input("확인할 월을 입력하시오."))  # 확인할 월 입력
if(1 <= month <= 12) :  # 1~12 값인지 확인
    day = int(input("확인할 일을 입력하시오."))  # 확인할 일 입력
    if(1 <= day <= 31) :  # 1~31 값인지 확인
        if(month > 3) : # 입력 월이 3월 이상일 경우 28 count에 추가
            count += 28
            if(month in day_31) : # 입력 월이 day_31 리스트에 있을 경우
                count += 31 * day_31.index(month)
                count += 30 * (day_30.index(month-1) + 1)
            elif(month in day_30) : # 입력 월이 day_30 리스트에 있을 경우
                count += 30 * day_30.index(month)
                count += 31 * (day_31.index(month-1) + 1)
        elif(month == 3) : # 입력 월이 3월일 경우 1월과 2월 일수 count에 추가
            count += (31 + 28)
        elif(month == 2) : # 입력 월이 2월일 경우 1월 일수 count에 추가
            count += 31
        count += day # 입력 일 count에 추가
        week_day = count % 7 # 출력할 요일 선택
        print("2018년 {}월 {}일은 {}요일입니다.".format(month, day, day_of_week[week_day-1]))
    else :
        print("일(day)을 잘못 입력하셨습니다.")
else :
    print("월(month)을 잘못 입력하셨습니다.")
```

클래스로 변경한 코드

```
class day_of_week :
    def __init__(self, month, day, day_30, day_31, day_list) : # 생성자 메소드
        self.month = month
        self.day = day
        self.day_30 = day_30
        self.day_31 = day_31
        self.day_list = day_list
    def january_march(self) : # 1월에서 3월의 경우 일수 반환 메소드
        if(self.month == 1) :
            return self.day
        elif(self.month == 2) :
            return self.day + 28
        elif(self.month == 3) :
            return self.day + 28 + 31
    def remains(self) : # 1월에서 3월을 제외한 나머지 월의 일수 반환 메소드
        count = 0
        if(self.month > 3) : # 입력된 월이 3월보다 클 경우 2월의 일수 추가
            count += 28
        if(self.month in self.day_31) : # 입력된 월이 31일인 월일 경우 이전에 있던 30일, 31일 월수 추가
            count += 31 * self.day_31.index(self.month)
            count += 30 * (self.day_30.index(self.month - 1) + 1)
        elif(self.month in self.day_30) : # 입력된 월이 30일인 월일 경우 이전에 있던 30일, 31일 월수 추가
            count += 30 * self.day_30.index(self.month)
            count += 31 * (self.day_31.index(self.month - 1) + 1)
        return count
    def final_count(self) : # 구한 일수를 최종적으로 구하여 7로 나눈 나머지 반환 메소드
        count = 0
        if(1 <= self.month <= 3) : # 입력된 월이 1월에서 3월 사이인 경우 구한 일수 반환
            count = self.january_march()
        else : # 이외의 나머지 월일 경우 입력된 일수 추가
            count = self.remains() + self.day
        return count % 7
    def to_string(self) : # 결과 출력 메소드
        return "{}월 {}일은 {}요일입니다.".format(self.month, self.day, self.day_list[self.final_count() - 1])
day_30 = [4, 6, 9, 11]
day_31 = [1, 3, 5, 7, 8, 10, 12]
day_week = ['월', '화', '수', '목', '금', '토', '일']
weeks = [
    day_of_week(3, 3, day_30, day_31, day_week),
    day_of_week(6, 28, day_30, day_31, day_week),
    day_of_week(4, 24, day_30, day_31, day_week)
]
for week in weeks :
    print(week.to_string())
```

(3) 결과 화면

```
확인할 월을 입력하시오.3
확인할 일을 입력하시오.3
2018년 3월 3일은 토요일입니다.
```

(4) 예외처리 화면

(4-1) 1~12월 이외의 값을 입력한 경우

```
확인할 월을 입력하시오.16
월(month)을 잘못 입력하셨습니다.
```

(4-2) 1~31일 이외의 값을 입력한 경우

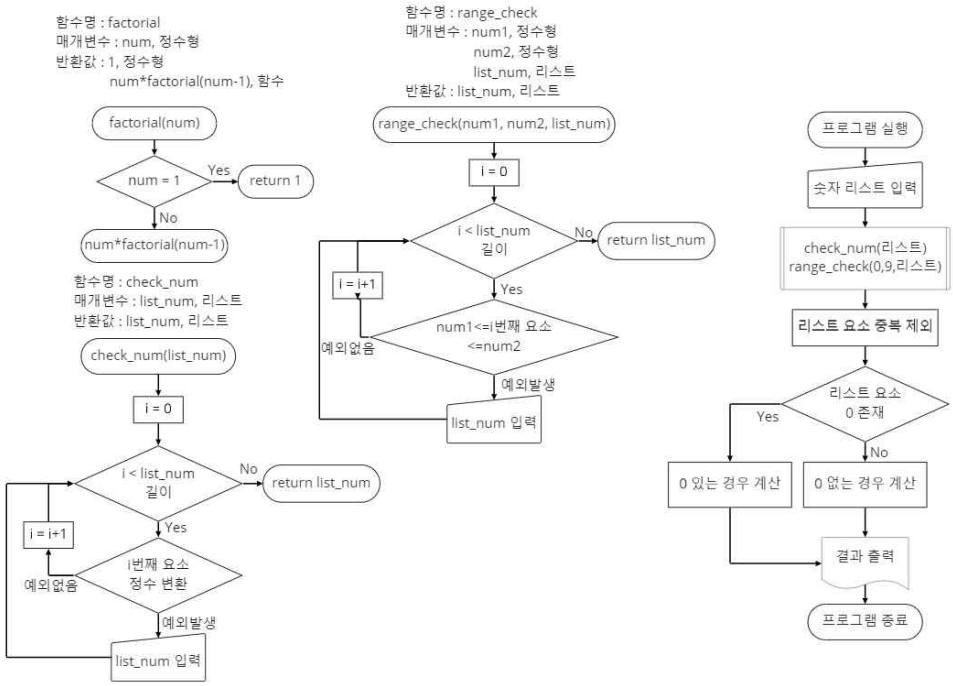
```
확인할 월을 입력하시오.12
확인할 일을 입력하시오.32
일(day)을 잘못 입력하셨습니다.
```

(5) 설명

월, 일을 입력하여 2018년의 해당 날짜 요일을 출력하는 프로그램입니다.
2018년 1월 1일은 월요일인 것을 기준으로 입력받은 날짜의 요일을 계산합니다.
1~12월, 1~31일 이외의 값을 입력한 경우 경고문을 출력하고 프로그램을 종료합니다.
클래스의 경우 day_of_week(월, 일, 30일 월, 31일 월, 요일 리스트) 형식으로 만들었습니다.

9) 숫자 나열하는 경우의 수 구하기

(1) Flow Chart



(2) 소스코드

```
def factorial(num) : # 팩토리얼 함수
    if num == 1 :
        return 1
    else :
        return num * factorial(num-1)
def check_num(list_num) : # 리스트 요소들이 숫자인지 확인하는 함수
    while True :
        try :
            for i in range(len(list_num)) : # 요소를 정수로 변환하여 리스트 반환
                int(list_num[i])
            return list_num
        except : # 에러 발생시 재입력
            print("잘못 입력하셨습니다. 숫자를 입력해주십시오.")
            list_num = input("다시 입력해주십시오. : ").split()
def range_check(num1,num2,list_num) : # 요소가 범위 안 값인지 확인하는 함수
    while True :
        for i in range(len(list_num)) : # 범위 안 값이 아닐 경우 재입력
            if(int(list_num[i]) < num1 or int(list_num[i]) > num2) :
                print("{}에서 {}사이의 숫자를 입력해주십시오.".format(num1, num2))
                list_num = input("다시 입력해주십시오. : ").split()
        return list_num
# 리스트에 0이 있는지 체크합니다. 기본값은 False 입니다.
zero_exisit = False
input_num = input("0에서 9사이의 숫자를 3개 이상 입력해주십시오. : ").split() # 배열할 숫자 입력
input_num = check_num(input_num) # 숫자 확인 함수 호출
input_num = range_check(0,9,input_num) # 제한 범위 확인 함수 호출
input_num = list(set(input_num)) # 숫자 중복 제외 후 리스트 변환
for i in range(len(input_num)) : # 입력값 중 0이 있는지 확인
    if(input_num[i] == "0") :
        zero_exisit = True
if(zero_exisit == True) : # 0이 있을 경우 계산
    result = (len(input_num) - 1) * factorial(len(input_num) - 1)
else : # 0이 없을 경우 계산
    result = factorial(len(input_num))
print("{}를 나열할 수 있는 경우의 수는 총 {}가지 입니다.".format(input_num, result))
```

(3) 결과 화면

```
0에서 9사이의 숫자를 3개 이상 입력해주십시오. : 1 9 1
['9', '1']를 나열할 수 있는 경우의 수는 총 2가지 입니다.
```

(4) 예외처리 화면

(4-1) 문자를 입력한 경우

```
0에서 9사이의 숫자를 3개 이상 입력해주십시오. : 1 9 팔
잘못 입력하셨습니다. 숫자를 입력해주십시오.

다시 입력해주십시오. :
```

(4-2) 범위를 벗어난 숫자를 입력한 경우

```
0에서 9사이의 숫자를 3개 이상 입력해주십시오. : 1 11 2
0에서 9사이의 숫자를 입력해주십시오.

다시 입력해주십시오. :
```

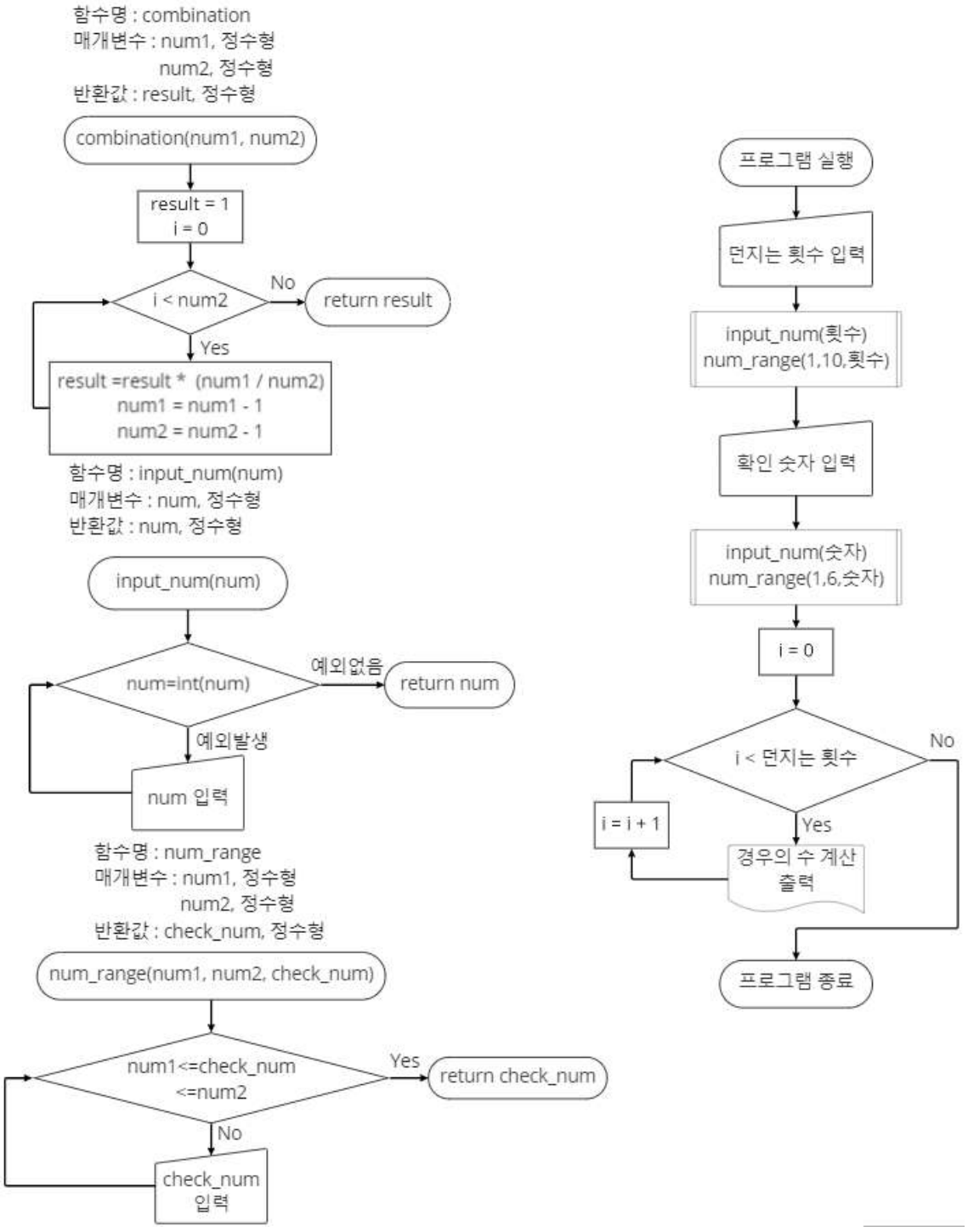
(5) 설명

3개 이상의 숫자를 입력받아 해당 숫자를 나열할 수 있는 경우의 수를 계산하는 프로그램입니다. 숫자 중복을 제외한 후 숫자 리스트에 0이 들어있는지 판별합니다. 0이 들어간 경우 맨 앞에 둘 수 없기 때문에 0이 있을 경우와 없을 경우를 구분해서 계산합니다.

check_num 함수, range_check 함수를 사용하여 입력 값이 숫자인지, 범위 안의 값이 맞는지 체크하고 에러 발생시 경고문을 출력하고 다시 입력 받습니다.

10) 주사위 n번 던져서 특정 번호가 나올 횟수별로 경우의 수 구하기

(1) Flow Chart



(2) 소스코드

기본 소스 코드

```
def combination(num1,num2) : # 확률의 조합 계산 함수, aCb의 값 출력
    result = 1
    for i in range(num2) :
        result *= (num1 / num2) # result에 num1/num2 곱하고 각각 1을 뺀 후 다시 곱하는 것 반복(num2가 1이 될 때까지)
        num1 -= 1
        num2 -= 1
    return int(result)
def input_num(num) : # 입력값이 숫자인지 확인하는 함수
    while True:
        try : # 입력값 정수 변환하고 반환
            num = int(num)
            return num
        except ValueError: # 에러 발생시 재입력
            print("잘못 입력하셨습니다. 숫자를 입력해주십시오.")
            num = input("다시 입력해주십시오.")
def num_range(num1,num2,check_num) : # 입력값이 제한된 범위인지 확인하는 함수
    while True :
        if (num1 <= check_num <= num2) : # 숫자가 제한된 범위 안의 값일 경우 반환
            return check_num
        else : # 제한 범위를 넘겼을 경우 input_num 함수 호출
            print("범위 안의 값을 입력해주십시오.")
            check_num = input("다시 입력해주십시오 : ")
            check_num = input_num(check_num)
times = input("주사위 던지는 횟수를 입력해주세요 : ") # 주사위 던질 횟수 입력
times = input_num(times) # 숫자 확인 함수 호출
times = num_range(1,10,times) # 제한 범위 확인 함수 호출
select_num = input("확인할 숫자를 입력해주세요 : ") # 확인할 숫자 입력
select_num = input_num(select_num) # 숫자 확인 함수 호출
select_num = num_range(1,6,select_num) # 제한 범위 확인 함수 호출
total_times = 6**times # 총 경우의 수 계산
print("총 경우의 수는 {}개 입니다.".format(total_times))
for i in range(times) : # 숫자 횟수별 경우의 수 계산, 출력
    extra_count = 5**(times-i-1)
    select_count = combination(times,i+1)
    count_list = extra_count * select_count
    print("{}가 {}번 나오는 경우는 {}번 입니다.".format(select_num,i+1,count_list))
```

클래스로 변경한 코드

```
def combination(num1,num2) : # 확률의 조합 계산 함수
    result = 1
    # aCb의 b가 1이 될 때까지 값을 계산합니다.
    for i in range(num2) :
        # result에 number1 / number2 의 값을 곱해주고 각각 1을 뺀 후 다시 곱해주는 것을 반복합니다.
        result *= (num1 / num2)
        num1 -= 1
        num2 -= 1
    return int(result)
class Number_Of_Cases : # 경우의 수 계산 클래스 선언
    def __init__(self, check_num, times) : # 생성자 메소드
        self.check_num = check_num
        self.times = times
    def total_count(self) : # 총 경우의 수 계산 메소드
        return "총 경우의 수는 {}개 입니다.".format(6**self.times)
    def result_string(self) : # 횟수별 경우의 수 출력 메소드
        for i in range(self.times) : # 경우의 수 계산
            result = 0
            result += combination(self.times,i+1) * (5**(self.times-1-i))
            print("{}가 {}번 나오는 경우는 {}번 입니다.".format(self.check_num,i+1,result))
        return 1
dice_counts = [
    Number_Of_Cases("1",4),
    Number_Of_Cases("5",3),
    Number_Of_Cases("2",2)
]
for dice in dice_counts:
    print(dice.total_count())
    print(dice.result_string())
```

(3) 결과 화면

```
주사위 던지는 횟수를 입력해주세요 : 3
확인할 숫자를 입력해주세요 : 1
총 경우의 수는 216개 입니다.
1가 1번 나오는 경우는 75번 입니다.
1가 2번 나오는 경우는 15번 입니다.
1가 3번 나오는 경우는 1번 입니다.
```

(4) 예외처리 화면

(4-1) 문자를 입력한 경우

```
주사위 던지는 횟수를 입력해주세요 : 삼
잘못 입력하셨습니다. 숫자를 입력해주십시오.

다시 입력해주십시오.
```

(4-2) 범위를 벗어난 숫자를 입력한 경우

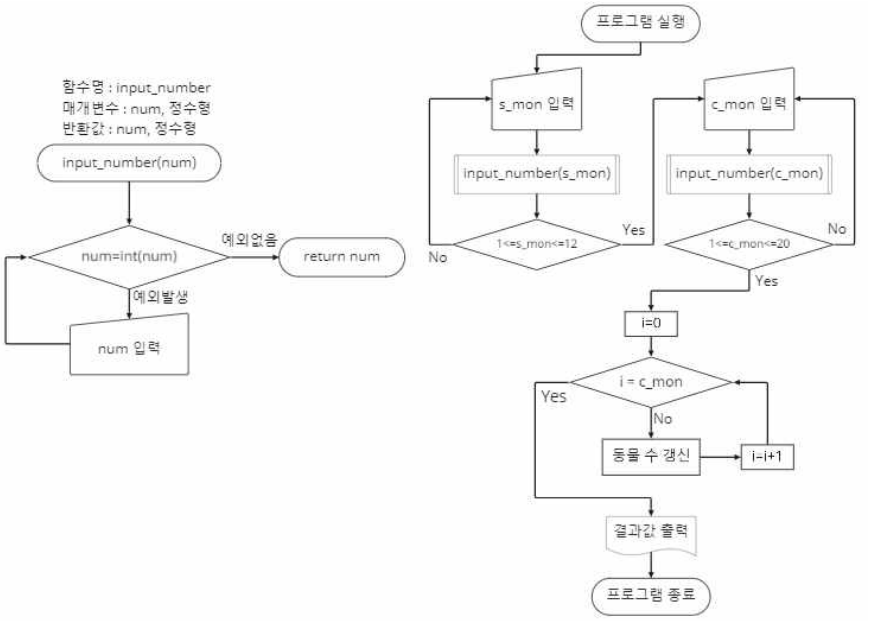
```
주사위 던지는 횟수를 입력해주세요 : 11
범위 안의 값을 입력해주십시오.

다시 입력해주십시오 :
```

(5) 설명

주사위를 n번 던졌을 때 숫자 x가 y번 나올 경우의 수를 계산하는 프로그램입니다.
n, x를 입력하면 y를 1~n번까지로 설정하여 경우의 수를 출력합니다.
input_num, num_range 함수를 사용하여 숫자인지, 범위 안의 값을 입력했는지 판별하고 오류 발생시 경고문을 출력하고 다시 입력합니다.
클래스의 경우 Number_Of_Cases(주사위 숫자, 횟수) 형식으로 만들었습니다.

11) 동물 번식 실험 결과 예측 프로그램

(1) Flow Chart



(2) 소스코드

```
def input_number(num): # 숫자를 입력했는지 확인하는 함수
    while True:
        try: # 정수 변환 가능시 변환하여 반환
            num = int(num)
            return num
        except ValueError: # 변환 불가능시 경고문과 재입력문 출력
            print("숫자를 입력해주십시오.")
            num = input("다시 입력해주십시오 : ")

while True:
    s_mon = input("동물을 풀어주는 달을 골라주세요 : ") # 동물을 풀어주는 달 입력
    s_mon = input_number(s_mon) # 숫자 입력 확인 함수 호출
    if (1 <= s_mon <= 12):
        break
    else:
        print('1~12월 중에서 입력해주세요')
while True:
    c_mon = input("몇 달 뒤에 확인하시겠습니까?(1~20 중 선택) : ") # 확인할 개월 수 입력
    c_mon = input_number(c_mon) # 숫자 입력 확인 함수 호출
    if(1 <= c_mon <= 20): # 확인하는 개월수 1~20으로 제한
        break
    elif(c_mon == 0): # 0을 입력할 경우
        print("동물을 처음 풀어준 달입니다.")
    else: # 이외의 값을 입력한 경우
        print('범위에 맞는 숫자를 입력해주세요')
# 초기값 입력
adult_animal = 0
kid_animal_zero = 2
kid_animal_one = 0
if check_month:
    for i in range(c_mon): # 개월수만큼 반복하여 증가하는 동물 수를 계산합니다.
        adult_animal += kid_animal_one
        kid_animal_one = kid_animal_zero
        kid_animal_zero = adult_animal
total_animal = adult_animal + kid_animal_zero + kid_animal_one # 최종 동물 수를 대입
n_mon = (s_mon + c_mon) % 12 # 12로 나눈 나머지로 12월 이후 다시 1월부터 시작
print("{}월에 동물을 풀어줬고 {}달 뒤에 확인합니다.".format(s_mon,c_mon))
print("{}월에 확인한 동물의 수는 {}마리 입니다.".format(n_mon,total_animal))
```

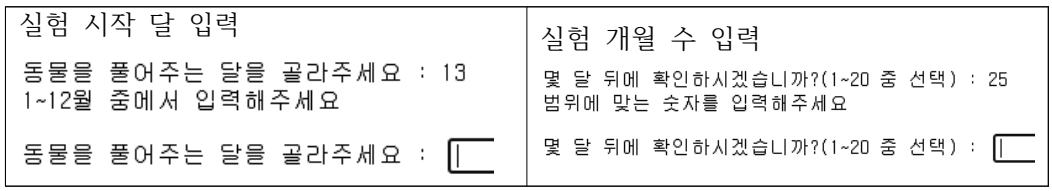
(3) 결과 화면

```
동물을 풀어주는 달을 골라주세요 : 1
몇 달 뒤에 확인하시겠습니까?(1~20 중 선택) : 10
1월에 동물을 풀어줬고 10달 뒤에 확인합니다.
11월에 확인한 동물의 수는 178마리 입니다.
```

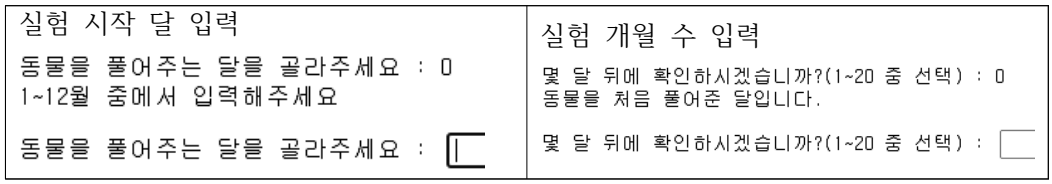
(4) 예외처리 화면

(4-1) 범위를 벗어난 숫자, 0, 음수를 입력한 경우

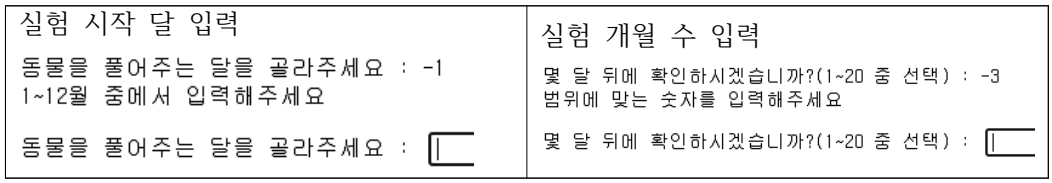
- 범위 벗어난 경우



- 0 입력한 경우

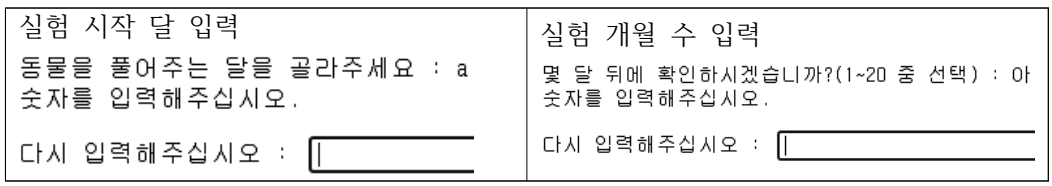


- 음수 입력한 경우



범위를 벗어나거나 0, 음수를 입력할 경우 제한 범위를 출력하고 다시 입력 받습니다.

(4-2) 문자 입력한 경우



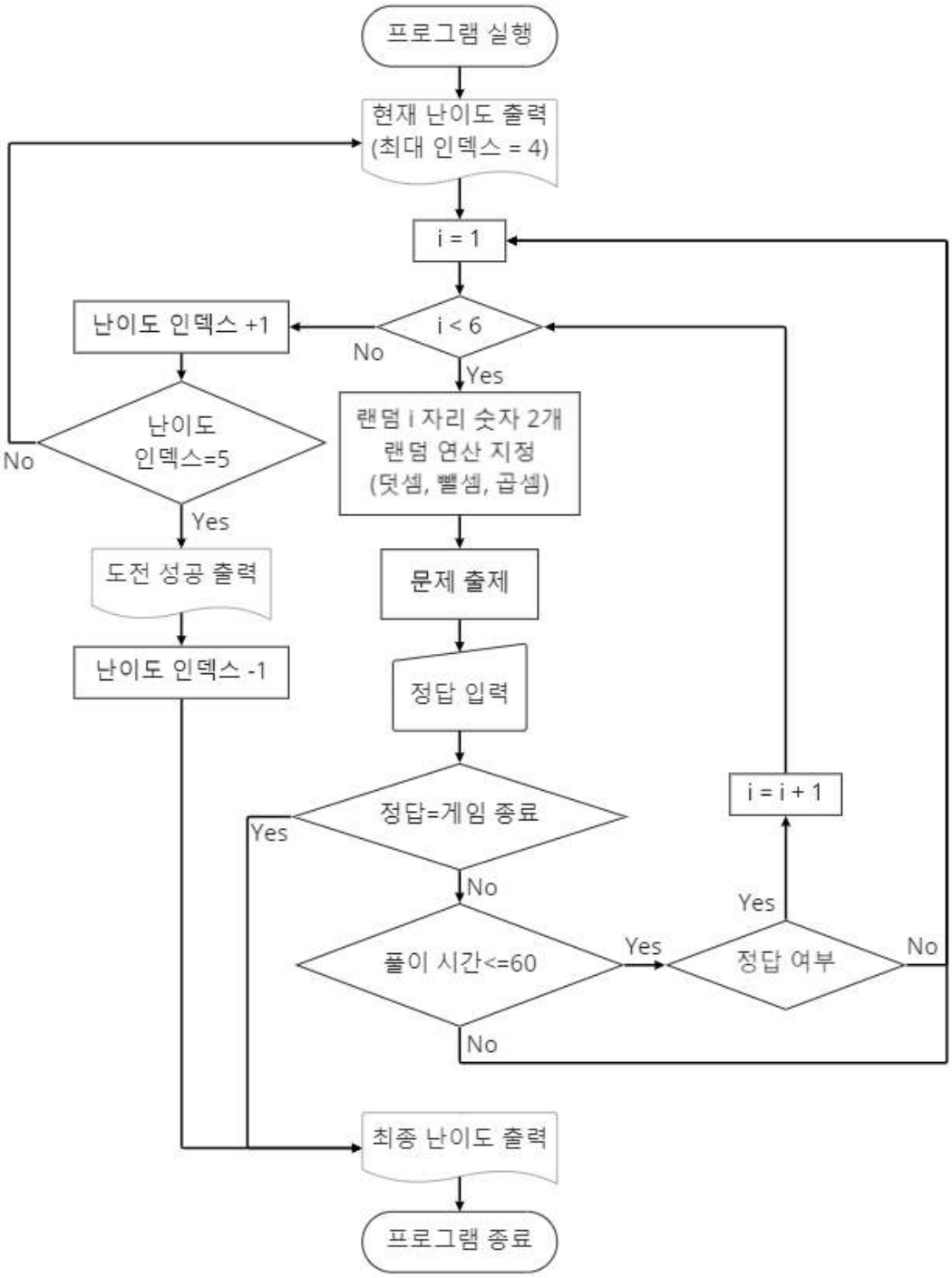
경고문을 출력하고 다시 입력 받습니다.

(5) 설명

동물을 풀어줄 달을 입력받고 실험을 진행할 개월 수를 입력받아 결과를 출력합니다.
태어난 동물, 1개월 된 동물, 성체가 된 동물로 나눠 개월마다 갱신하여 총 동물 수를 계산합니다.
범위를 벗어나거나 0, 음수, 문자를 입력한 경우 다시 입력받도록 하였습니다.

12) 연산 문제 생성기 프로그램

(1) Flow Chart



(2) 소스코드

```
import random
import time

difficulty_list = ['VERY EASY', 'EASY', 'NORMAL', 'HARD', 'VERY HARD'] # 연산 난이도 리스트
arithmetic_list = ['+', '-', 'X'] # 연산 방법 리스트
play_game = '게임 시작'
difficulty_cnt = 0 # 연산 난이도
while play_game == '게임 시작' :
    game_cnt = 1
    print('현재 난이도는 {}입니다.'.format(difficulty_list[difficulty_cnt])) # 현재 난이도 출력
    random_range = 10 ** difficulty_cnt # 난이도에 따라 숫자 자리수 변경
    while game_cnt < 3 :
        number1 = random.randrange(random_range,random_range * 10)
        number2 = random.randrange(random_range,random_range * 10)
        arithmetic = random.randrange(0,3) # 연산 방법 랜덤 선택
        if arithmetic == 0 : # 더하기
            result = number1 + number2
        elif arithmetic == 1 : # 빼기
            result = number1 - number2
        elif arithmetic == 2 : # 곱하기
            result = number1 * number2
        print('{} {} {} = '.format(number1, arithmetic_list[arithmetic], number2))
        s_time = time.time() # 문제 풀이 시작 시간
        user_answer = input('정답을 입력해주십시오 : ')
        e_time = time.time() # 문제 풀이 종료 시간
        t_time = e_time-s_time # 문제 풀이 걸린 시간
        if user_answer != '게임 종료' :
            if t_time <= 60 : # 걸린 시간 60초 이하
                if int(user_answer) == result or float(user_answer) == result :
                    game_cnt += 1
                    if game_cnt <= 2 : # 5문제 맞추는 조건
                        print('맞습니다. 소요시간 : {: .0f}초 \n{}번째 문제입니다.'.format(t_time, game_cnt))
                    else : # 난이도 변경
                        print('2문제를 모두 맞혔습니다. 난이도를 변경합니다.')
                        print('-'*50)
                        difficulty_cnt += 1
                else : # 문제를 틀린 경우 난이도 반복
                    game_cnt = 1
                    print('틀렸습니다. 다시 %s 난이도를 반복합니다.' %difficulty_list[difficulty_cnt])
            else : # 걸린 시간 초과
                print('제한시간을 초과하였습니다. 다시 %s 난이도를 반복합니다.' %difficulty_list[difficulty_cnt])
        else :
            game_cnt = 6
            play_game = user_answer
        if difficulty_cnt == 5 :  # 마지막 난이도까지 완수한 조건
            print('도전 성공!')
            difficulty_cnt -= 1
            play_game = '게임 종료'
print('게임이 종료되었습니다. 도달한 난이도는 {}입니다.'.format(difficulty_list[difficulty_cnt]))
```

(3) 결과 화면

결과 화면1

```
현재 난이도는 VERY EASY입니다.
9 X 9 =
정답을 입력해주십시오 : 81
맞습니다. 소요시간 :  2초
2번째 문제입니다.
9 X 5 =
정답을 입력해주십시오 : 45
2문제를 모두 맞혔습니다. 난이도를 변경합니다.
--------------------------------------------------
현재 난이도는 EASY입니다.
23 X 65 =
정답을 입력해주십시오 : 1495
맞습니다. 소요시간 :  8초
2번째 문제입니다.
87 X 11 =
정답을 입력해주십시오 : 957
2문제를 모두 맞혔습니다. 난이도를 변경합니다.
--------------------------------------------------
현재 난이도는 NORMAL입니다.
850 - 566 =
정답을 입력해주십시오 : 284
맞습니다. 소요시간 :  7초
2번째 문제입니다.
128 X 920 =
정답을 입력해주십시오 : 117760
2문제를 모두 맞혔습니다. 난이도를 변경합니다.
--------------------------------------------------
```

결과 화면2

```
현재 난이도는 HARD입니다.
5187 - 5096 =
정답을 입력해주십시오 : 91
맞습니다. 소요시간 :  10초
2번째 문제입니다.
4803 - 2820 =
정답을 입력해주십시오 : 1983
2문제를 모두 맞혔습니다. 난이도를 변경합니다.
--------------------------------------------------
현재 난이도는 VERY HARD입니다.
62338 - 60799 =
정답을 입력해주십시오 : 1539
맞습니다. 소요시간 :  11초
2번째 문제입니다.
22783 - 95312 =
정답을 입력해주십시오 : -72529
2문제를 모두 맞혔습니다. 난이도를 변경합니다.
--------------------------------------------------
도전 성공!
게임이 종료되었습니다. 도달한 난이도는 VERY HARD입니다.
```

(4) 예외처리 화면

(4-1) 정답을 틀린 경우

```
현재 난이도는 VERY EASY입니다.
3 X 4 =
정답을 입력해주십시오 : 12
맞습니다. 소요시간 :  4초
2번째 문제입니다.
6 - 6 =
정답을 입력해주십시오 : 1
틀렸습니다. 다시 VERY EASY 난이도를 반복합니다.
3 X 9 =

정답을 입력해주십시오 : |
```

(4-2) 제한 시간을 넘긴 경우

```
2번째 문제입니다.
6 + 5 =
정답을 입력해주십시오 : 11
제한시간을 초과하였습니다. 다시 VERY EASY 난이도를 반복합니다.
5 + 6 =

정답을 입력해주십시오 : |
```

(5) 설명

연산 문제를 생성하는 프로그램입니다.

난이도는 VERY EASY, EASY, NORMAL, HARD, VERY HARD로 총 5개이고 각 난인도마다 나오는 숫자 자리수가 정해집니다. 연산은 덧셈, 뺄셈, 곱셈이 있습니다.

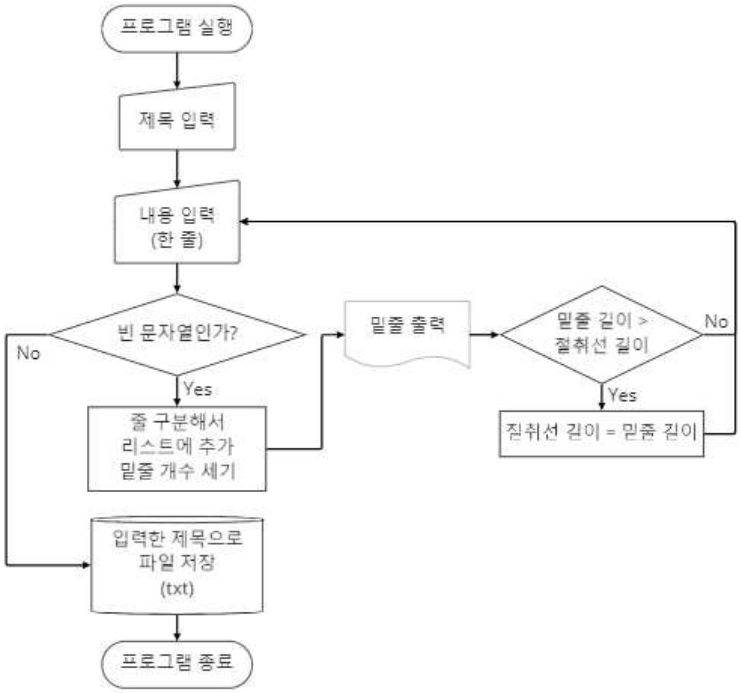
문제를 연속해서 5번 맞출 때 난이도가 상승하며 중간에 1번이라도 틀리면 다시 연속해서 5문제를 맞춰야 합니다. 제한 시간 60초를 넘긴 경우는 틀린 것으로 간주합니다.

VERY HARD 난이도에서 5문제를 연속으로 맞춘 경우 도전 성공 출력문을 띄우고 프로그램을 종료합니다. 문제를 푸는 도중 정답에 '게임 종료'를 입력한 경우에는 현재 난이도를 출력하고 프로그램을 종료합니다.

2. 파일 저장하기

1) 문서 작성 및 저장하는 프로그램

(1) Flow Chart



(2) 소스코드

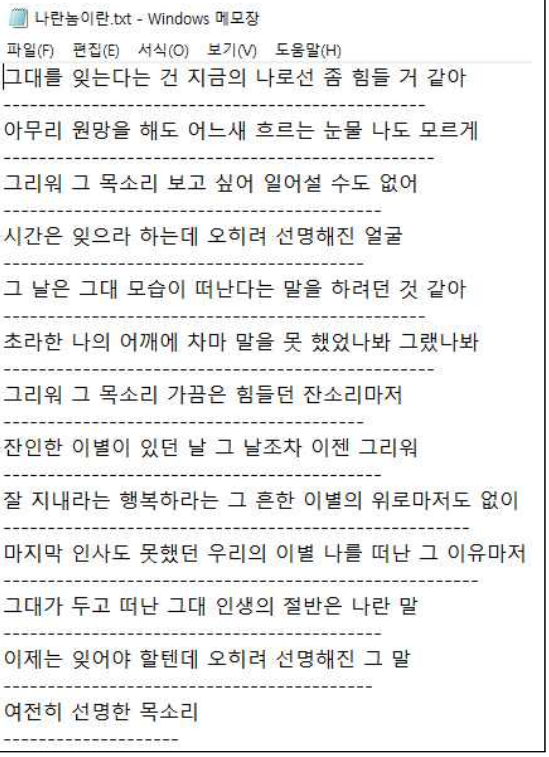
```
start = True
str_list = []
max_count = 0
input_title = input('제목 : ') # 제목 입력
while start :
    count = 0
    input_content = input('') # 내용 입력
    if input_content != '' :
        str_list.append(input_content + '₩n') # 한 줄 단위로 내용 리스트 정리
        input_content_list = list(input_content)
        for i in range(len(input_content_list)) : # 밑줄 길이 계산
            if input_content_list[i] == ' ' :
                count += 1
            else :
                count += 2
        print('-' * count)
        str_list.append('-' * count + '₩n')
        if count > max_count :  # 절취선 길이 갱신
            max_count = count
    else : # 공백 입력시 절취선 출력, 문서 작업 종료
        print('=' * max_count)
        start = False

# 지정된 경로에 리스트 요소를 반복하면서 텍스트 파일로 저장합니다.
f_name = './' + input_title + '.txt'
f = open(f_name, 'a', encoding='UTF-8') # 문서 저장
for i in range(len(str_list)) :
    f.write(str(str_list[i]))
f.close()
```

(3) 결과 화면

코드 결과 화면

```
제목 : 나란놈이란
그대를 잊는다는 건 지금의 나로선 좀 힘들 거 같아
----------
아무리 원망을 해도 어느새 흐르는 눈물 나도 모르게
----------
그리워 그 목소리 보고 싶어 일어설 수도 없어
----------
시간은 잊으라 하는데 오히려 선명해진 얼굴
----------
그 날은 그대 모습이 떠난다는 말을 하려던 것 같아
----------
초라한 나의 어깨에 차마 말을 못 했었나봐 그랬나봐
----------
그리워 그 목소리 가끔은 힘들던 잔소리마저
----------
잔인한 이별이 있던 날 그 날조차 이젠 그리워
----------
잘 지내라는 행복하라는 그 흔한 이별의 위로마저도 없이
----------
마지막 인사도 못했던 우리의 이별 나를 떠난 그 이유마저
----------
그대가 두고 떠난 그대 인생의 절반은 나란 말
----------
이제는 잊어야 할텐데 오히려 선명해진 그 말
----------
여전히 선명한 목소리
----------

==========
```

저장 결과 화면



(4) 설명

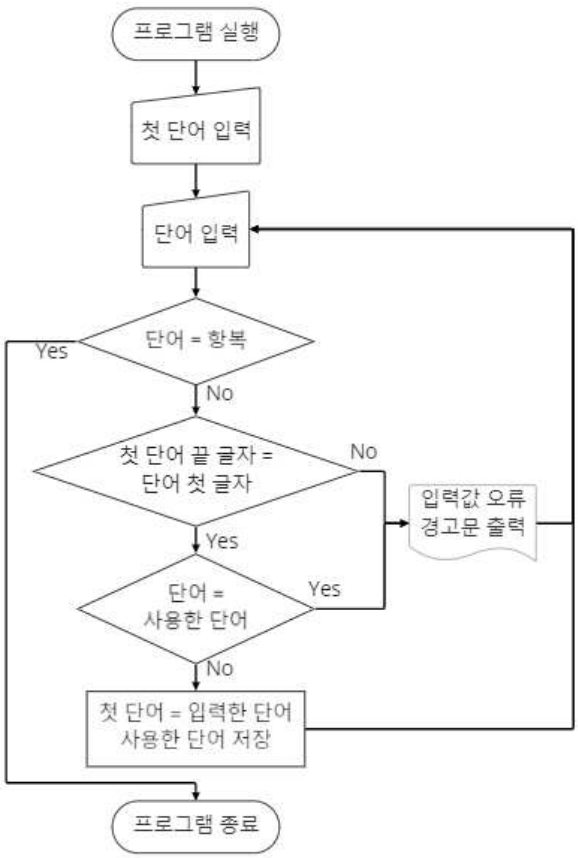
입력한 그대로 문서로 저장하는 프로그램입니다.

문장을 입력하면 그 길이에 맞춰 밑줄을 생성하고 리스트에 밑줄까지 포함하여 저장합니다. 공백을 입력할 경우 입력을 종료하며 상대경로로 제목을 파일 이름으로 저장합니다.

3. 게임 프로그램

1) 끝말잇기 게임

(1) Flow Chart



(2) 소스코드

```
input_str = 'start'
word_list = []
first_str = input('첫 단어를 입력해주십시오. : ')  # 첫 단어 입력
first_str_list = list(first_str)  # 리스트 변환
while input_str == 'start' :
    next_str = input('단어를 입력해주십시오. : ')  # 단어 입력
    if next_str != '항복' :
        next_str_list = list(next_str)
        if first_str_list[-1] == next_str_list[0] :
            if next_str not in word_list : # 사용한 단어 리스트와 비교
                first_str_list = next_str_list  # 시작 단어 교체
                word_list.append(next_str)  # 사용한 단어 리스트에 추가
            else : # 사용한 단어를 입력한 경우
                print('이미 사용한 단어입니다.')
        else : # 시작 단어를 잘못 입력한 경우
            print('시작 단어가 잘못 되었습니다.')
    else : # 항복 입력시 게임 종료
        print('게임을 종료합니다.')
        input_str = 'end'
```

(3) 결과 화면

```
첫 단어를 입력해주십시오. : 비트코딩
단어를 입력해주십시오. : 딩동댕유치원
단어를 입력해주십시오. : 원자력발전소
단어를 입력해주십시오. : 소말리아
단어를 입력해주십시오. : 아르헨티나
단어를 입력해주십시오. : 나이지리아
단어를 입력해주십시오. : 아프가니스탄
단어를 입력해주십시오. : 탄자니아
단어를 입력해주십시오. : 아라비아반도
단어를 입력해주십시오. : 항복
게임을 종료합니다.
```

(4) 예외처리 화면

(4-1) 이전 단어의 끝 단어를 사용하지 않은 경우

```
첫 단어를 입력해주십시오. : 비트코딩
단어를 입력해주십시오. : 딩동댕유치원
단어를 입력해주십시오. : 자전거
시작 단어가 잘못 되었습니다.

단어를 입력해주십시오. :
```

(4-2) 사용한 단어를 다시 입력한 경우

```
첫 단어를 입력해주십시오. : 비트코딩
단어를 입력해주십시오. : 딩동댕유치원
단어를 입력해주십시오. : 원자력발전소
단어를 입력해주십시오. : 소말리아
단어를 입력해주십시오. : 아르헨티나
단어를 입력해주십시오. : 나이지리아
단어를 입력해주십시오. : 아프가니스탄
단어를 입력해주십시오. : 탄자니아
단어를 입력해주십시오. : 아프가니스탄
이미 사용한 단어입니다.

단어를 입력해주십시오. :
```

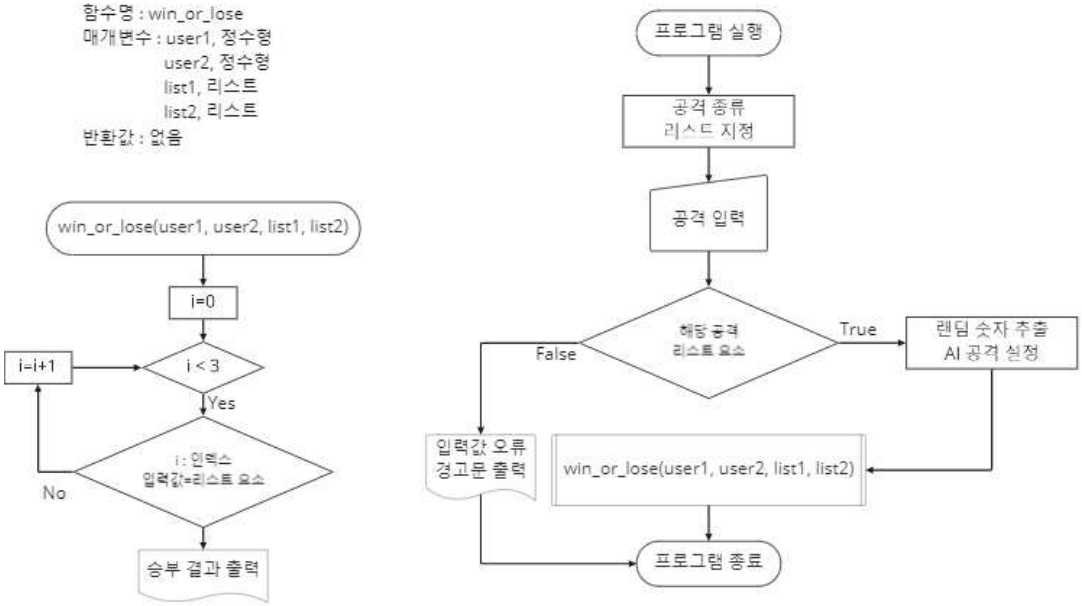
(5) 설명

끝말잇기 게임을 하는 프로그램입니다.

처음 단어를 입력하면 해당 단어의 끝 단어로 시작하는 단어를 입력해야 합니다. 단어를 맞게 입력한 경우 해당 단어를 사용한 단어 리스트에 추가합니다. 시작 단어가 틀리거나 사용한 단어 리스트에 있는 단어를 입력한 경우 경고문을 출력하고 다시 입력 합니다. 항복을 입력한 경우 프로그램을 종료합니다.

2) 가위바위보 게임

(1) Flow Chart



(2) 소스코드

```python
import random  # random 모듈을 호출합니다.

def win_or_lose(user1,user2,list1,list2): # 승부 판별 함수
    for i in range(3):
        if(user1 == list1[i] or user1 == list2[i]): # 입력한 값 리스트에서 탐색
            if(list1[user2] == list1[i]): # 나와 AI가 비길 때
                print("비겼습니다.")
            elif(list1[user2] == list1[(i+1)%3]): # AI가 이길 때
                print("AI가 이겼습니다.")
            else: # AI가 졌을 때
                print("당신이 이겼습니다.")

# 리스트를 정리합니다.
list_object1 = ['가위', '바위', '보']
list_object2 = ['찌', '묵', '빠']
user = input("나 : ") # 내가 낼 공격 입력
if (user in list_object1 or user in list_object2): # 입력이 리스트에 있을 경우
    ai = random.randrange(0,3) # 랜덤 숫자를 가져와 AI의 공격 설정
    print("상대 : {}".format(list_object1[ai]))
    win_or_lose(user,ai,list_object1,list_object2) # 승부 판별 함수 호출
else: # 가위바위보와 다른 수를 입력했을 경우 예외 처리
    print("가위, 바위, 보 또는 찌, 묵, 빠 중에서 입력해주십시오.")
```

(3) 결과 화면

```
나 : 가위
상대 : 가위
비겼습니다.
```

(4) 예외처리 화면

가위, 바위, 보 또는 찌, 묵, 빠가 아닌 단어를 입력한 경우

```
나 : 바가
가위, 바위, 보 또는 찌, 묵, 빠 중에서 입력해주십시오.
```

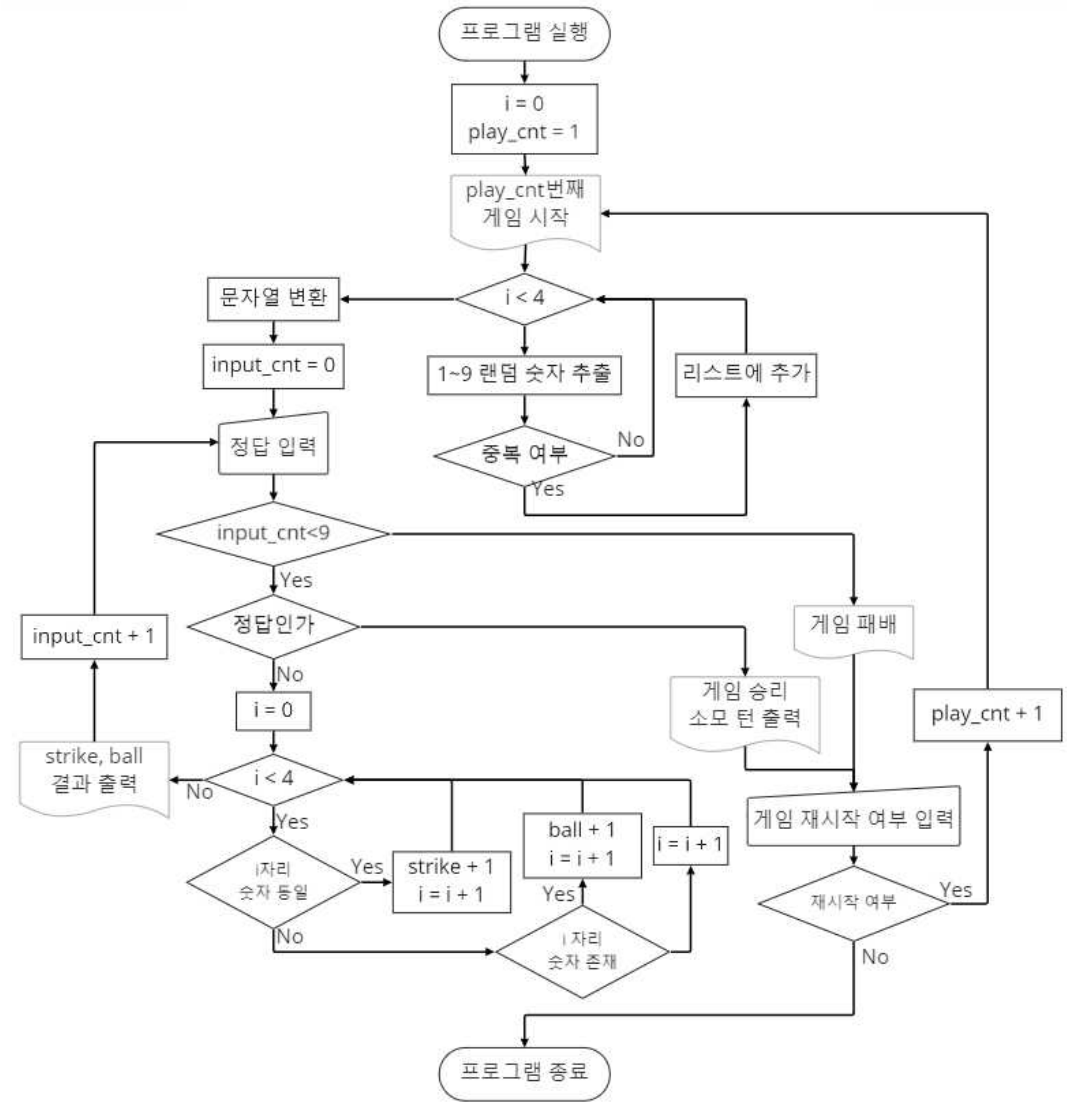
(5) 설명

AI와 가위바위보 게임을 하는 프로그램입니다.

user는 자신이 낼 공격을 입력하고 AI는 랜덤으로 0~2 숫자를 가져와 공격을 정합니다. 해당 공격에 따라 승, 패, 비기는 경우를 출력합니다.

3) 숫자야구 게임

(1) Flow Chart



(2) 소스코드

```
import random

replay = True
play_cnt = 1
while replay == True :
    winner = 'computer'
    input_cnt = 0
    print('%s번째 게임을 시작합니다.' %play_cnt) # 현재 실시한 게임 횟수 출력
    com_number_list = []
    while len(com_number_list) < 4 : # AI가 랜덤 숫자 4개 지정
        random_number = random.randint(1,9)
        if random_number not in com_number_list : # 숫자 중복 제외
            com_number_list.append(random_number)
    for i in range(len(com_number_list)) : # 요소 문자 변환 후 문자열로 변환
        com_number_list[i] = str(com_number_list[i])
    com_number = ''.join(com_number_list)
    print('AI가 숫자를 정했습니다.')
    print(com_number)
    while input_cnt < 9: # 정답 제출 기회 9번
        strike_cnt = 0
        ball_cnt = 0
        user_number = input('AI가 생각한 숫자는 무엇일까요? :') # 정답 입력
        input_cnt += 1
        user_number_list = []
        for i in user_number : # 각 숫자 리스트 요소로 정리
            user_number_list.append(i)
        if user_number != com_number : # 정답이 아닐 경우
            for i in range(len(com_number_list)) : # 자리와 숫자가 맞을 경우 strike
                if user_number_list[i] == com_number_list[i] :
                    strike_cnt += 1
                else : # 자리는 다르고 숫자는 맞을 경우 ball
                    if user_number_list[i] in com_number_list :
                        ball_cnt += 1
            print('strike : {}, ball : {}'.format(strike_cnt, ball_cnt)) # 결과 출력
        else : # 정답인 경우, user 승리
            print('숫자를 맞추셨습니다.')
            print('%s 턴을 소모하셨습니다. 게임에서 승리하셨습니다.' %input_cnt)
            winner = 'user'
            break
    if winner == 'computer' : # AI 승리
        print('사용 가능한 턴을 모두 소모하셨습니다. 게임에서 패배하셨습니다.')
    while True:
        replay = input('다시 게임을 시작하시겠습니까?(Y / N) :') # 재경기 여부
        replay = replay.upper()
        if replay == 'Y' or replay == 'YES' :
            play_cnt += 1
            print('-'*40)
            replay = True
            break
        elif replay == 'N' or replay == 'NO':
            replay = False
            break
        else :
            print('Y 또는 N를 입력해주십시오.')
print('게임을 종료합니다.')
```

(3) 결과 화면

게임 승리 결과 화면

```
1번째 게임을 시작합니다.
AI가 숫자를 정했습니다.
AI가 생각한 숫자는 무엇일까요? :1234
strike : 0, ball : 2
AI가 생각한 숫자는 무엇일까요? :6789
strike : 1, ball : 1
AI가 생각한 숫자는 무엇일까요? :1256
strike : 0, ball : 2
AI가 생각한 숫자는 무엇일까요? :1257
strike : 0, ball : 1
AI가 생각한 숫자는 무엇일까요? :1258
strike : 1, ball : 1
AI가 생각한 숫자는 무엇일까요? :6138
strike : 2, ball : 1
AI가 생각한 숫자는 무엇일까요? :6418
숫자를 맞추셨습니다.
7 턴을 소모하셨습니다. 게임에서 승리하셨습니다.

다시 게임을 시작하시겠습니까?(Y / N) :
```

게임 패배 결과 화면

```
1번째 게임을 시작합니다.
AI가 숫자를 정했습니다.
AI가 생각한 숫자는 무엇일까요? :1234
strike : 0, ball : 2
AI가 생각한 숫자는 무엇일까요? :5678
strike : 0, ball : 1
AI가 생각한 숫자는 무엇일까요? :1256
strike : 0, ball : 0
AI가 생각한 숫자는 무엇일까요? :3478
strike : 0, ball : 3
AI가 생각한 숫자는 무엇일까요? :4378
strike : 1, ball : 2
AI가 생각한 숫자는 무엇일까요? :4783
strike : 0, ball : 3
AI가 생각한 숫자는 무엇일까요? :4873
strike : 0, ball : 3
AI가 생각한 숫자는 무엇일까요? :4387
strike : 1, ball : 2
AI가 생각한 숫자는 무엇일까요? :3487
strike : 0, ball : 3
사용 가능한 턴을 모두 소모하셨습니다. 게임에서 패배하셨습니다.

다시 게임을 시작하시겠습니까?(Y / N) :
```

대소문자 구분 없이 yes 또는 y 입력

```
다시 게임을 시작하시겠습니까?(Y / N) :y
----------------------------------------
2번째 게임을 시작합니다.
AI가 숫자를 정했습니다.

AI가 생각한 숫자는 무엇일까요? :
```

대소문자 구분 없이 no 또는 n 입력

```
8 턴을 소모하셨습니다. 게임에서 승리하셨습니다.
다시 게임을 시작하시겠습니까?(Y / N) :n
게임을 종료합니다.
```

(4) 예외처리 화면

재시작 여부 결정에서 대소문자 구분 없이 y, yes, n, no 이외의 값을 입력한 경우

```
다시 게임을 시작하시겠습니까?(Y / N) :네
Y 또는 N를 입력해주십시오.

다시 게임을 시작하시겠습니까?(Y / N) :
```

(5) 설명

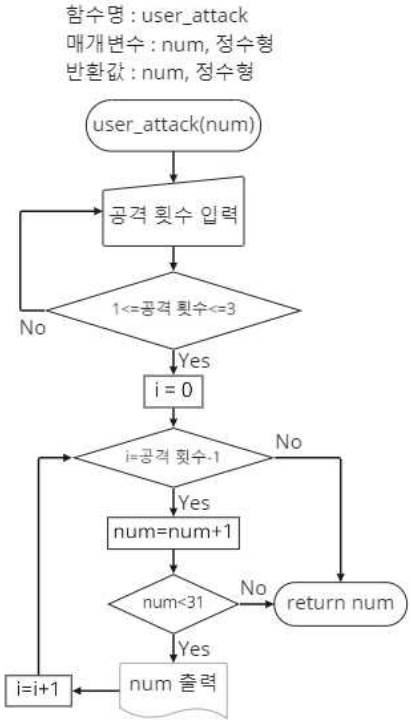
AI와 숫자야구를 하는 게임입니다.

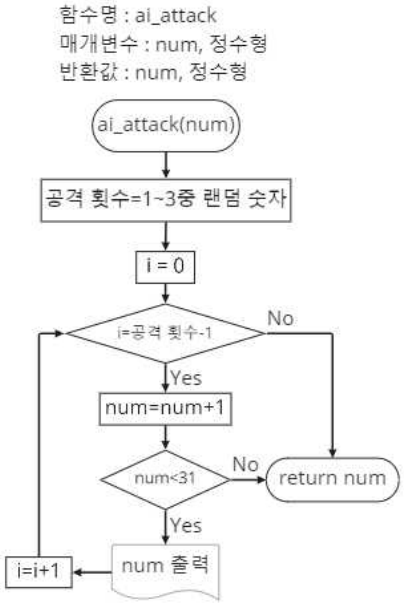
AI가 1~9 사이의 숫자를 랜덤으로 가져온 후 숫자 리스트에 추가합니다. 이때 리스트에 중복된 숫자가 있으면 추가하지 않습니다. 리스트 길이가 4가 될 때까지 이 작업을 반복한 후 리스트 숫자를 문자열로 바꿉니다.

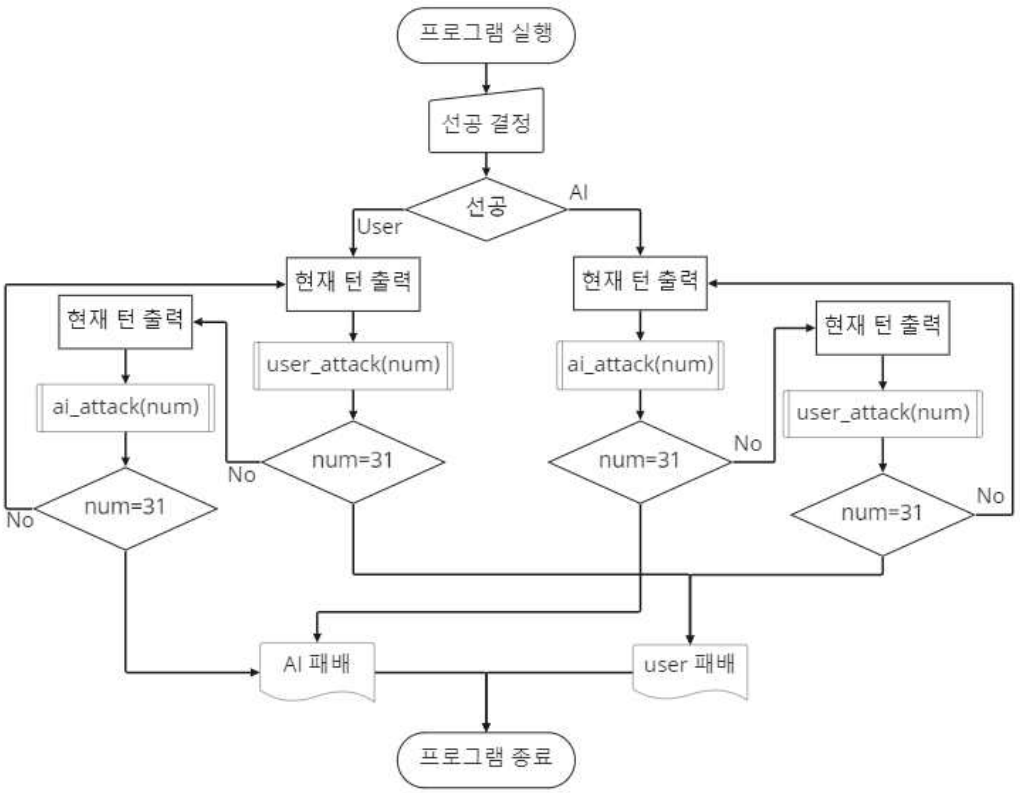
user는 AI가 정한 숫자가 무엇인지 입력합니다. 입력 기회는 최대 9번까지이고 자리와 숫자가 모두 맞을 경우 strike 숫자만 맞을 경우 ball로 카운트하여 출력합니다. 9번의 기회 동안 정답을 말하지 못할 경우 user의 패배이고 승리의 경우 소모한 턴 수를 출력합니다. 게임 종료 후 게임 재시작 여부를 물으며 대소문자 구분 없이 y, yes, n, no로 동작하도록 하였고 이외의 문자 입력시 다시 입력 합니다.

4) 베스킨 라빈스 31

(1) Flow Chart







(2) 소스코드

```python
import random # random 모듈을 호출합니다.
def user_attack(num): # 입력한 횟수만큼 숫자를 출력하는 함수(user용)
    while True:
        print("당신차례 입니다.")
        times = int(input(f'{num+1}부터 몇개를 외치겠습니까?')) # 시작 숫자 출력, 횟수 입력
        if(1 <= times <= 3): # 1~3 안의 값을 입력할 경우
            for i in range(times):
                num += 1
                if(num < 31): # 출력 값이 31보다 작을 경우 출력, 1 증가
                    print("당신 {}".format(num))
                elif(num == 31): # 출력 값이 31일 경우 종료
                    print("당신 {}".format(num))
                    break
            return num
        else: # 입력 범위를 벗어난 경우
            print("1부터 3사이의 숫자를 입력해주십시오.")

def ai_attack(num): # 입력한 횟수만큼 숫자를 출력하는 함수(AI용)
    print("AI 차례입니다.")
    times = random.randrange(1,4) # 랜덤으로 1~3 사이의 값으로 횟수 설정
    print("AI가 {}부터 {}번 외칩니다.".format(num+1,times)) # 시작 숫자, 횟수 출력
    for i in range(times):
        num += 1
        if(num < 31): # 출력 값이 31보다 작을 경우 출력, 1 증가
            print("AI {}".format(num))
        elif(num == 31): # 출력 값이 31일 경우 종료
            print("AI {}".format(num))
            break
    return num
# 초기값 설정
turn_count = 0
number = 0
while True:
    attack_order = input("당신부터 하시겠습니까? (yes/no) : ").upper() # 선공 결정
    if attack_order in ['YES', 'Y', 'NO', 'N']: # 대소문자 관계 없이 YES, Y, NO, N 이외의 값은 다시 입력
        break
    else:
        print('잘못 입력하셨습니다. 다시 입력해주세요.')
if(attack_order == 'YES' or attack_order == 'Y'): # user 선공
    while(0 <= number <= 31):
        turn_count += 1
        print("<{} 턴>----------------------".format(turn_count)) # 현재 턴 출력
        number = user_attack(number) # 함수 호출, user 공격
        if number == 31: # 31을 말하고 종료된 경우 패배선언
            print("당신이 졌습니다.")
            break
        turn_count += 1
        print("<{} 턴>----------------------".format(turn_count)) # 현재 턴 출력
        number = ai_attack(number) # 함수 호출, AI 공격
        if(number == 31): # 31을 말하고 종료된 경우 패배선언
            print("AI가 졌습니다.")
            break
elif(attack_order == 'NO' or attack_order == 'N'): # AI 선공
    while(0 <= number <= 31):
        turn_count += 1
        print("<{} 턴>----------------------".format(turn_count)) # 현재 턴 출력
        number = ai_attack(number) # 함수 호출, user 공격
        if number == 31: # 31을 말하고 종료된 경우 패배선언
            print("AI가 졌습니다.")
            break
        turn_count += 1
        print("<{} 턴>----------------------".format(turn_count)) # 현재 턴 출력
        number = user_attack(number) # 함수 호출, AI 공격
        if(number == 31): # 31을 말하고 종료된 경우 패배선언
            print("당신이 졌습니다.")
            break
```

(3) 결과 화면

결과 화면1

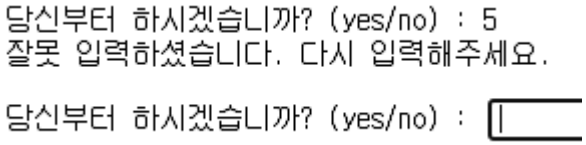
```
당신부터 하시겠습니까? (yes/no) : y
<1 턴>----------------------
당신차례 입니다.
1부터 몇개를 외치겠습니까?3
당신 1
당신 2
당신 3
<2 턴>----------------------
AI 차례입니다.
AI가 4부터 3번 외칩니다.
AI 4
AI 5
AI 6
<3 턴>----------------------
당신차례 입니다.
7부터 몇개를 외치겠습니까?3
당신 7
당신 8
당신 9
<4 턴>----------------------
AI 차례입니다.
AI가 10부터 3번 외칩니다.
AI 10
AI 11
AI 12
<5 턴>----------------------
당신차례 입니다.
13부터 몇개를 외치겠습니까?3
당신 13
당신 14
당신 15
<6 턴>----------------------
AI 차례입니다.
AI가 16부터 3번 외칩니다.
AI 16
AI 17
AI 18
```

결과 화면2

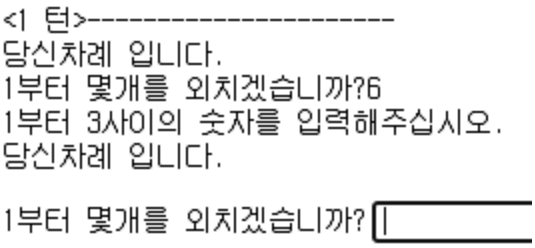
```
<7 턴>----------------------
당신차례 입니다.
19부터 몇개를 외치겠습니까?3
당신 19
당신 20
당신 21
<8 턴>----------------------
AI 차례입니다.
AI가 22부터 1번 외칩니다.
AI 22
<9 턴>----------------------
당신차례 입니다.
23부터 몇개를 외치겠습니까?3
당신 23
당신 24
당신 25
<10 턴>----------------------
AI 차례입니다.
AI가 26부터 1번 외칩니다.
AI 26
<11 턴>----------------------
당신차례 입니다.
27부터 몇개를 외치겠습니까?3
당신 27
당신 28
당신 29
<12 턴>----------------------
AI 차례입니다.
AI가 30부터 3번 외칩니다.
AI 30
AI 31
AI가 졌습니다.
```

(4) 예외처리 화면

(4-1) 선공 결정에서 대소문자 관계없이 YES, Y, NO, N 이외의 값 입력할 경우



(4-2) 횟수 입력에서 1~3 이외의 값을 입력한 경우



(5) 설명

AI와 베스킨 라빈스31 게임을 하는 프로그램입니다.

게임 규칙은 1부터 시작해서 숫자를 연속으로 부르고 마지막에 31을 부른 사람이 패배합니다. 숫자는 한 번에 3개까지 말할 수 있고 건너뛰기는 불가능합니다.

사용자는 자신이 부를 횟수를 입력하고 AI는 자동으로 설정하여 게임을 진행합니다.

선공 결정에서 대소문자 관계없이 YES, Y, NO, N 입력으로 선공을 결정하게 만들었고 이외의 값을 입력할 경우 다시 입력합니다.

사용자가 횟수를 입력할 때 1~3의 범위를 넘어서는 숫자를 입력하면 다시 입력합니다.