프로그래밍 언어 개론 1주차 과제

과목명	프로그래밍 언어개론[00]
담당교수님	조은선 교수님
이름	정성욱
학번	201404376

(1) Double factorial에 대한 해결 방법:

Double_factorial의 함수를 정의하고 long값 하나를 입력을 받습니다.
이 Double_factorial에 대한 매커니즘에 대해 알아보겠습니다.
입력값을 하나 받습니다. n이라 가정하겠습니다.
들어온 값이 0이냐 1이나 1일 경우 1을 무조건 1을 내보냅니다.
0이나 1이 아닐 경우, 새로운 변수 n_n을 만들어 n값에서2를 뺍니다.
그리고 함수를 반환하면서 반환 동시에 Double_factorial을 호출하는데 이때
n(초기 입력값)을 곱해주는 것을 반환합니다.(Double_factorial(n_n)*n)
이런식이게 되면 10을 넣었다고 가정합시다.
f(10*f(8*f(6*f(4*f(2*f(1)))))) 인데 1*2*4*6*8*10입니다.

(2) Faery Sequence 에 대한 해결 방법:

해결을 위해 제가만든 몇 가지 핵심적인 함수의 기능들을 보겠습니다.
*Compare: 분모리스트, 분자리스트를 인자로 받습니다. 여기서 각 리스트에서 각 분자/분모 값을 뽑아서 계산하고 비교해서 오름차순으로 정렬하는 역할을 합니다.
*print_list: 분모리스트, 분자리스트를 인자로 받습니다. 각 리스트에서 분자/분모의 형태로 하나씩 뽑아서 출력합니다.

이제 문제 해결을 위한 proceed_fs 함수에 대해 보겠습니다.
간단한 풀이원리를 보자면 저는, 페리수열의 분수 자체를 저장하기보다 분모,분자 나누어서 따로따로 저장해두고 필요할 때 하나씩 불러와서 처리하는 식으로 짰습니다.
인자로 (ArrayList p,ArrayList c,int begin, int end) 이렇게 받는데, p는 분모 리스트들 c는 분자 리스트입니다. begin은 현재 몇 번째 페리인지 나타내는 변수입니다. end는 유저가 입력으로 주는 수인데 몇 번 페리수열을 수행할지 언제 끊을지 정하는 기준입니다.

시작하면 내부에 페리 연산을 돕기 위해 분모,분자 리스트들의 인덱스값들을 임시로 저장하기 위한 index Arraylist를 만듭니다. 분자, 분모 역시 를 하나씩 만들어 줍니다.(이들의 변수명은 index:인덱스 리스트,c_value:분자 리스트,p_value:분모 리스트) 그리고 앞서 매개변수로 받았던 begin과 end에 대한 분기 조건문을 만드는데, 이것은 아래에 페리 수열 갱신 연산이 시행되기 전에 함수를 리턴합니다. 처음에서 사용자가 입력했던 n(end)값과 현재의 몇 번째로 호출되었는지 개수를 세는 begin의 값을 비교하여 end보다 begin이 클 경우 그전에 탈출 합니다.

if문을 들어가지 않았다면, 초기에 만들어두었던 임시index와 p_value,c_value에 값을 넣기 위한 작업을 합니다. 그러나 넣기 전에 루프를 돌리는데 현재 페리수열이 가지고 있는 크기만큼 루프를 돌리면서 측정할 것입니다. 여기에는 for루프를 도는 I와 J가있습니다. 그리고 끝을 일부러 페리 크기-1 만큼 두고 돌렸습니다. J=I+1로 설정했기 때문에 항상 I보다 앞서는데 이것을 이용해서 수열 전체의 리스트를 비교하면서 다음 페리수열 새로 생겨날 변수들을 바보셈 계산합니다. 이러한 계산 중에는 몇가지 버려야하는 값들도 나온는데 그런 값을 처리하기 위한 방법으로 바보셈으로 나온 분모의 값이 현재의 F값, 즉 현재가 F3을 계산중인데 분모가 3보다 크다면, c_value랑 p_value에 넣지 않고 컨티뉴를 합니다. 그렇게 된다면 현재 수열에서 어디에(index) 어떤값이(c_value/p_value)가 생길지가 정해집니다. 이 값들을 가지고 기존 리스트에 바보셈한 결과를 넣어야하는데 하기전에 한가지 가정을 해봅시다. 기존의 수열이 [0/1, 1/2 ,1/1] 이라고 쳤을 때 추가되는 수의 인덱스 리스트는 [1,2],분모 [3,3],분자 [1,2] 이라고 가정합시다. 그랬을 때 제 로직을 '한'번만 수행해보겠습니다. 기존 수열은 [0/1, 1/3 ,1/2 ,1/1]이 될것이고, index[2] 분자[2] 분모[3] 남을 것인데, 다음에 이것을 그대로 한번 더 수행 해버리면 문제가 생깁니다. 1/3 1/2 사이에 2/3 값이 들어가는데 그것을 방지하기 위해 idx_count=0으로 두어 로직이 한번 수행될 때마다 하나씩 증가시키고 매번 index값을 넣을 때 넣어줍니다. 그러면 큰 에러없이 잘 수행됩니다. 물론 지금까지 로직으로하면 큰 문제는 되지는 않았지만 Compare는 혹시나 모를 로직의 오류로 인해 문제가 생길까봐 버그 생기지 말라고 안전빵으로 한번더 크기별로 소팅하는 함수입니다. 그리고 소팅이 끝난후에 콘솔에 바보셈이 끝난 페리수열을 출력합니다. 후에 보조 리스트들을 초기화 시키고 다시 proceed_fs를 호출 시키는데 매개변수로 새롭게 편성된 페리수열 분모 분자를 주고 현재가 그다음 호출이라는 것을 알리기위해 begin을 1더하고 end는 그대로 보내줍니다.(=> proceed_fs(p,c,++begin,end);)

(3)느낀 점

1번 문제는 간단하게 풀었는데, 2번이 조금 까다로웠습니다. 처음에는 그냥 배열로 하려다가 메모리 변동이 심해서 데이터의 오고감에 있어서 자유로운 리스트를 써야겠다 생각하고 풀었습니다. 알고리즘을 못해서 그냥 있는 공식 그대로를 구현하자 하는 생각으로 코드를 짰는데 코드가 좀 복잡해지고, 빅오도 적지않은 편이라 아쉬움이 남지만, 설계한 대로 풀려서 한편으로는 뿌듯했습니다. 처음 과제내용들을 때 에이 8시간 걸리겠어? 했는데 막상 해보니 그거보다 약간 더 걸렸는데, 지금 생각해도 왜 그렇게 걸렸는 지 잘 모르겠네요 ㅎㅎ

(4)구현 결과

```
1번문제 더블 팩토리얼 input:100

output:342432247025119762482464328952081859751186750537191988279156544634880000000000000
2번문제 페리수열 input:8
f1:[0/1, 1/1]
f2:[0/1, 1/2, 1/1]
f3:[0/1, 1/3, 1/2, 2/3, 1/1]
f4:[0/1, 1/4, 1/3, 1/2, 2/3, 3/4, 1/1]
f5:[0/1, 1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 1/1]
f6:[0/1, 1/6, 1/5, 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 1/1]
f7:[0/1, 1/7, 1/6, 1/5, 1/4, 2/7, 1/3, 2/5, 3/7, 1/2, 4/7, 3/5, 2/3, 5/7, 3/4, 4/5, 5/6, 6/7, 1/1]
f8:[0/1, 1/8, 1/7, 1/6, 1/5, 1/4, 2/7, 1/3, 3/8, 2/5, 3/7, 1/2, 4/7, 3/5, 5/8, 2/3, 5/7, 3/4, 4/5, 5/6, 6/7, 7/8, 1/1]
```