Programming Language Homework 3

資訊 107 F74036069 徐佳筠

1. Introduction

I. First, I ask the user to input the number, and store the number by `read(X)`. Then I call `goldbach(X)` to find the goldbach combination of this input. `halt` makes the program end when the whole task is completed.

II. In the `goldbach(X)` function, I will determine the following:
   i. Whether the number is even, by `isEven(X)`.
   ii. Whether the number is smaller or equal then 2, by `X<=2`.

   If the input is an odd number or not greater than 2, it will print `invalid input.`, and terminates the program.
   If the input satisfies the conditions, it will call `findResult(A, B)`, and look for the combinations from `2` and `X-2`.

III. `isEven(X)` find the even numbers by determining the remainder of the input divided by 2. If the remainder is 0, it is even. If not, it is odd.

IV. The `findResult(A, B)` function does the following:
   i. Since we only need to find the combinations, we only need to search for the possibly until we reach the midpoint.
   ii. Then I assign `X` to `A`, and `Y` to `B`, and check if they are prime

numbers respectively. If one of them is not a prime number, then we will look for the next one, by using `findResult(A + 1, B - 1)`.

The reason that I assign `X` to `A`, and `Y` to `B` instead of using `A` and `B`, is that I found that `A` and `B` would store the process of how I get that number, instead of the number itself (as shown in the picture below). Not only will this make it harder to read, but cause error in some cases.

```
Input : 12.
Output :
2+1+1+1 12-2-1-1-1
2+1+1+1+1+1 12-2-1-1-1-1-1
```

iii. If both `A` and `B` are prime numbers, then the program will print out this combination, and look for the next possible combination by `findResult(A + 1, B - 1)`.

V. I use `isPrime(2).` to define that 2 is a prime. For the other numbers greater than 2, we will find if it is divisible by any other numbers beside itself and 1 by `divisible(X, Y)`.

VI. I start checking whether the number `X` is divisible by 2 in `divisible(X, Y)`. If not, then I will keep looking for the next possible number until the square of `Y` is greater than `X` or it is divisible by a certain number.

2. Result

```
C:\Users\Kathy\Desktop\goldbach>swipl -q
1 ?- [goldbach].
Input : 55.
Output :
invalid input.

C:\Users\Kathy\Desktop\goldbach>swipl -q
1 ?- [goldbach].
Input : 100.
Output :
3 97
11 89
17 83
29 71
41 59
47 53
```