프로그래밍 언어론 과제02

201511449

컴퓨터과학과

김동현

1. 아래 코드에서 보인 for과 while 반복문은 완벽하게 똑같이 동작하지는 않는다. 일치하지 않는 예를 보인다. (힌트: continue 사용 부분을 생각해볼 것)

for (i = first; i <= last; i += step) {

…

}

{

i = first;

while (i <= last) {

…

i += step;

}

}

위의 for문은 first~last범위를 last-first횟수만큼 step씩 증가 한다.

(단, 중간에 break로 반복문을 빠져나가지 않을 때)

i = first;

while (i <= last) {

…

i += step;

}

}

while문은 아무런 조건이 없을 때는 위의 for문 과 같이 반복을 하지만 i+=step위에 continue문을 달아 i가 특정 숫자일 때 i+=step을 실행 하지 않으면 i<=last을 만족 못 할 때 까지의 loop의 반복횟수가 증가합니다.

예시) for (i = first; i <= last; i += step) {

if(i%3==1)

continue;

}

{

i = first;

while (i <= last) {

if(i%3==1)

continue;

i += step;

}

}

1. 일반적인 recursion에 비해 tail-recursion의 장점이 무엇인지 설명하시오.

재귀가 느려지는 이유는 일을 끝마치고 돌아갔을 때, 해야 할 일이 남아있기 때문이다.

컴파일러가 꼬리 재귀를 인식하는 경우에는 함수가 재귀적으로 호출될 때 리턴되는 위치를 스택에 저장하지 않는다.

일반 재귀함수 예시

int foo ( int n )

{

if ( n == 0 )

{

return 1;

}

return 2 \* foo ( n - 1 );

}

꼬리재귀(tail recursion)함수 예시

int gcd ( int m, int n )

{

if ( n == 0 )

{

return m;

}

else

{

return gcd ( n, m%n );

}

}