Homework 1

2014313303, 홍태하

1. **Make a segment of any c code to show that it is a good language.**

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Calculate.c

#include <stdio.h>

#include “function.h”

int main(){

printf(“%d\n”,sum(4,5));

printf(“%d\n”,minus(5,3));

return 0;

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

function.h

int sum(int, int);

int minus(int, int);

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

funcSum.c

#include “function.h”

int sum(int n, int m){

return n+m;

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

funcMinus.c

#include “function.h”

int minus(int n, int m){

return n-m;

}

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

하나의 프로그램을 만드는데 3개의 C파일과 하나의 헤더파일로 구성하였다. 실제로 큰 프로그램을 만들다 보면 코드량도 많아지고 여러 사람이 분업해서 작업을 할 경우가 있다. 이런 경우에 C언어는 각각의 사람이 코드를 만들고 그것들을 헤더파일을 사용해서 엮을 수 있게 한다. 따라서 소프트웨어를 부품화하여 간단하게 가져다 쓸 수 있다는 장점이 있다.

#include <stdio.h>

void swap(int\* a, int\* b);

int main(){

int n1 = 14, n2 = 25;

printf(“n1 : %d, n2 : %d\n”,n1,n2);

swap(&n1,&n2);

printf(“n1 : %d, n2 : %d\n”,n1,n2);

return 0;

}

void swap(int\* a, int\* b){

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

위 코드의 실행 결과는

n1 : 14, n2 : 25

n1 : 25, n2 : 14

이다. c언어의 또 하나의 장점은 포인터를 써서 어떤 변수의 메모리 주소에 접근할 수 있다는 것이다. 메모리 주소에 접근함으로써 함수 안에서 parameter의 값에 변화를 줄 수가 있다.

1. **Show the process how linear bound automata to check if an input string is palindrome.**

T : a,b

N : S, A,B,C

P :

S->Aa

A->aBC

aB->aC

C->bb

Terminal set에 a와b가 있고 Nonterminal set에 S와 A와 B와 C가 있는 linear bound automata이다.

P를 따라가면 S가 Aa가 되고 A가 aBC이기 때문에 Aa가 aBCa가 된다. 또 aB가 aC이기 때문에 aBCa는 aCCa가 된다. C는 bb이므로 aCCa는 abbbba가 된다. abbbba는 palindrome이므로 palindrome이 생성되는 process를 보였다.

S->Aa->aBCa->aCCa->abbbba

1. **Show two grammars which generate the same language.**

<identifier> ::= <letter>| <identifier><letter> | <identifier><digit>

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<letter> ::= A|B|C|…|X|Y|Z

위와 같이 BNF grammar를 정의하면 밑의 과정에 따라 ABC의 결과를 얻을 수 있다.

<identifier>-> <identifier><letter>

-> <identifier><letter><letter>

-> <letter><letter><letter>

-> A<letter><letter>

-> AB<letter>

-> ABC

<identifier> ::= <letter>| <identifier><letter>

<letter> ::= A|B|C|…|X|Y|Z

위와 같이 BNF grammar를 정의하면 밑의 과정에 따라 ABC의 결과를 얻을 수 있다.

<identifier>-> <identifier><letter>

-> <identifier><letter><letter>

-> <letter><letter><letter>

-> A<letter><letter>

-> AB<letter>

-> ABC

두 개의 다른 언어에서 같은 결과(ABC)를 보였다.