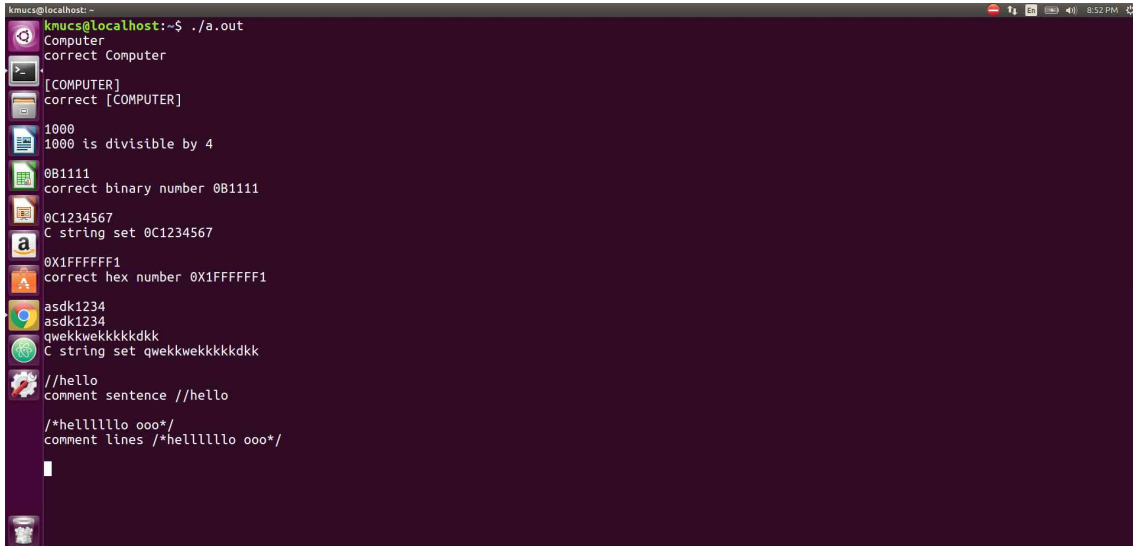


# 컴파일러 과제

20130940 장용훈

1



```
kmucs@localhost:~$ ./a.out
Computer
correct Computer
[COMPUTER]
correct [COMPUTER]
1000
1000 is divisible by 4
0B1111
correct binary number 0B1111
0C1234567
C string set 0C1234567
0X1FFFFFFF1
correct hex number 0X1FFFFFFF1
asdk1234
asdk1234
qwekkwekkkkkdkk
C string set qwekkwekkkkkdkk
//hello
comment sentence //hello
/*hellllllo ooo*/
comment lines /*hellllllo ooo*/
```

- Computer 와 같이 대문자로 시작해서 소문자로 이루어진 단어  
{Alpha}{alpha}+\$ 로 구성, 1개의 대문자 그리고 여러 개의 소문자로 끝날시 패턴이 매칭 되어 correct를 출력한다.

- [와 ] 사이에 대문자로만 구성된 단어  
"[{Alpha}+]"\$ “[로 시작하여 1개 이상의 대문자들과 ”로 끝 날 시에 패턴이 매칭되어 correct를 출력한다.

- 부호없는 이진수로 4의 배수인 수  
[0-1]+\$ 0과 1로 이루어진 이진수로 끝날시에 매칭되어 yytext를 atoi로 변환 시켜준후  
while (a>0){

```
16             if (a % 10==1)
17                 re += cnt;
18                 cnt *= 2;
19                 a /= 10;
20             }
```

다음과 같이 10진수로 변환한 뒤에 if(a%4==0) 조건을 만족하면 is divisible by 4라는 문장을 출력하게 된다.

- 0B로 시작되는 2진수  
"0B"[0-1]+\$을 조건으로하여 패턴 매칭시 correct를 출력.

- 0O로 시작되는 8진수  
"0O"[0-1]+\$을 조건으로하여 패턴 매칭시 correct를 출력.

- 0X로 시작되는 16진수

"0X"[0-1]+\$을 조건으로하여 패턴 매칭시 correct를 출력.

- C언어의 스트링 상수

[\40-\176]+\$ 아스키코드 값 40부터 176까지를 인식하여 매칭시 "C string set"이라는 문장을 출력해준다.

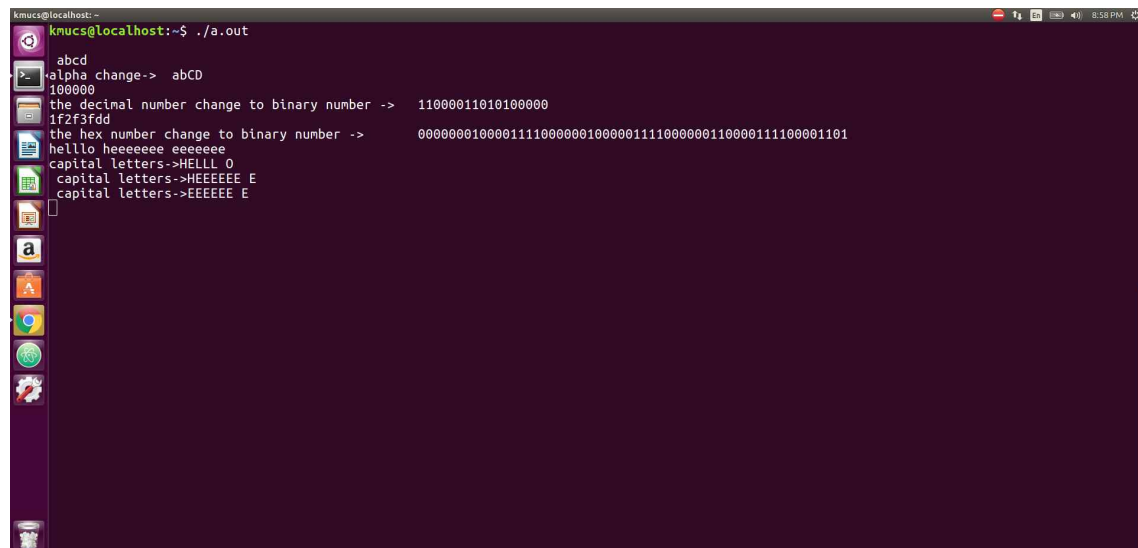
- //로 시작하여 라인끝까지 코멘트 문장

"/"[\40-\176]\*\$ //시작하고 뒤에 0개 이상의 아스키 코드 값이 온다면 매칭 성공하고 "comment lines"라는 문장을 출력한다.

- /\*와 \*/ 사이에 있는 코멘트 문장

"/"[\40-\176]\*"/\$ /\*시작하고 뒤에 0개 이상의 아스키 코드 값 그리고 끝에 \*/이 온다면 매칭 성공하고 "comment lines"라는 문장을 출력한다.

2.

A terminal window titled 'kmucs@localhost:~' showing the execution of a program. The prompt is 'kmucs@localhost:~\$ ./a.out'. The output is as follows:  
abcd  
alpha change-> abCD  
100000  
the decimal number change to binary number -> 11000011010100000  
1f2f3fdd  
the hex number change to binary number -> 00000001000011110000001000001111000000110000111100001101  
hello heeeeeee eeeeeee  
capital letters->HELLO  
capital letters->HEEEEEEE E  
capital letters->EEEEEE E

- 여러 개의 공백문자를 1개로 축약

```
for(int i=0; i<yyleng; i++)  
{  
    if(yytext[i]=='\t' | yytext[i]=='\32' | yytext[i]=='\n'){  
        else if (i==yyleng-1)printf(" %c",yytext[i]-32);  
        else printf("%c",yytext[i]-32);  
    }printf("\n");
```

다음과 같은 함수를 이용하여, 패턴이 매칭된 부분의 문자가 공백문자이면 출력하지 않고, 공백문자일시에만 출력해준다. 이때 마지막문자에는 SPACE를 해주고, 마지막에 \n을 하여 한 칸 띄어준다.

- tab 문자는 공백문자로 치환

[\t] {printf(" ")} 우선으로 하여, 탭문자가 매칭 될 시, SPACE를 출력한다.

- 10진수 또는 16진수를 2진수로 변환하여 출력 -- 함수 dtob() 사용

[0-9]+ 패턴 매칭시 dtob10을 이용하여 이진수로 변환하여 출력해준다.

[a-f|0-9]+ 패턴 매칭시 hexbin을 이용하여 16진수를 2진수로 변환하여 출력해준다.

- abcd의 cd를 CD로 치환

{alpha}{4} 4개의 알파벳을 매칭하여, 이때 매칭된 문자를 아래의 식을 이용한다.

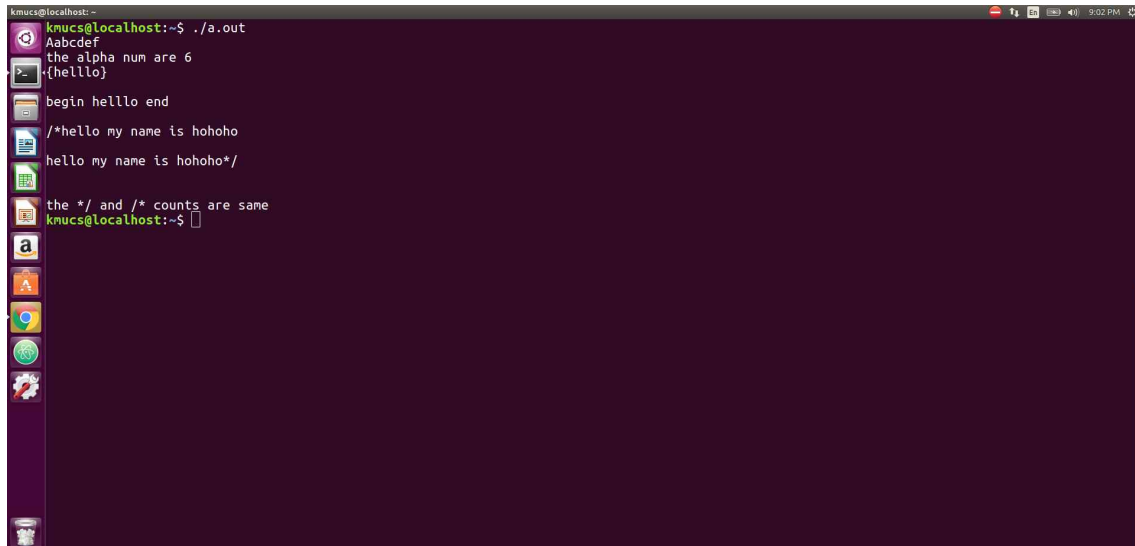
for(int i=2; i<yyleng; i++) printf("%c",yytext[i]-32); 매칭된 문자의 값에 32를 빼주면 아스키 코드값의 따라 대문자로 변하게 된다.

- 소문자를 모두 대문자로 치환

{alpha}+[\t\32]\*{alpha}\* 다음과 같이 중간의 공백까지 받아들이는 패턴이다.

printf("capital letters->");를 하여 바로 위의 문제와 같이 yytext[i]-32 처리를 하여 소문자를 대문자로 치환 해준다.

3



```
knucs@localhost:~$ ./a.out
Aabcdef
the alpha num are 6
{hello}
begin hello end
/*hello my name is hohoho
hello my name is hohoho*/
the */ and /* counts are same
knucs@localhost:~$
```

- 첫문자가 대문자이고 나머지는 모두 소문자인 단어의 개수를 count하여 출력

[A-Z]{1}[a-z]+ 한 개의 대문자 그리고 한 개 이상의 소문자로 이루어지면 매칭이 된다. 이때 for(int i=1; i<yyleng; i++) {count++;} 루프가 돌 때 마다, 카운트가 되기에, 총 소문자의 개수를 알게 된다. 그 후 the alpha num are을 출력해준다.

- /\*의 개수와 \*/의 개수를 count하고 개수가 같은지 비교 결과를 출력

\"/\*\"와 \"\*/\"를 각각 따로 만들어주어, 매칭 될 때 매다 카운트 해주고, 모든 입력이 끝났을 때 그 카운트 값을 비교하여 the \*/ and /\* counts are same 또는 the \*/ and /\* counts are different 라는 문장을 출력하게 된다.

- {는 begin으로, }는 end로 치환하여 출력

{\" {printf(\"begin \");} 과 \"} {printf(\" end\\n\");}를 각각 설정 해주어서, {가 매칭 될 시에는 begin }가 매칭될 시에는 end가 출력 되게 된다.

- 줄바꿈 문자를 모두 공백문자로 치환

[\\n]\* 줄바꿈 문자가 매칭될 시 , 그 횟수만큼 공백문자로 치환해준다. 그 부분은 다음과 같다.

```
{for(int i=0; i<yyleng; i++)printf(" "); }
```