```
//파일의 끝에 도달했는지 여부를 확인
//파일의 끝에 도달하지 못했을 경우에 0 반환
//파일 스트림의 끝을 만날 때까지 반복
while (!feof(filePtr))
{
    //정수 k는 0
    int k=0;

    //파일에서 한 줄 단위로 문자열을 읽어들인다.
    //filePtr의파일을 최대 4095수까지 읽고 line배열에 저장.
    fgets(line,4096,filePtr);

    // whil문 line[k]이 '\0'( null )이 아닐 때까지 반복.
    while(line[k]!='\0')
    {
        //  4-1) 만약 line[k]가 \t(탭키)일 시,
     if (line[k]=='\t')
     {
         //line[k]는 ‘ ‘으로 바꾼다.
        line[k]=' ';
     }
      //k에 1증가
     k++;
    }
    //5. line을 lineyedek에 문자열 복사
    strcpy(lineyedek,line);

    //curLine을 1증가
    curLine++;
    tempNode.val=-999; //tempNode.val값을 -999로 대입
    tempNode.exp_data=' '; //tempNode.exp_data값을 ‘‘로 대입
    tempNode.line=-999;//tempNode.line 을 -999로 대입
    tempNode.type=-999;//tempNode.type 을 -999로 대입


    // ⓐ하나
    //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
    //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
    //"begin"과 line이 동일하다면
    if (!strcmpi("begin\n",line) | !strcmpi("begin",line))
    {
        //foundMain == 0이면 False 0이 아니면 True
        //foundMain 이 1이면 if문 실행
        if (foundMain)
        {
            //tempNode.type 는 4이다.
            tempNode.type=4;
            ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
            STACK=Push(tempNode,STACK);
```

```
        }
    }


    //ⓑ
    // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
    //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
    //line이 "end"이거나 line이 "end \n"이라면
    else if (!strcmpi("end  \n",line) | !strcmpi("end",line) )
    {
        //(1)

        //foundMain == 0이면 False 0이 아니면 True
        //foundMain 이 1이면 실행
        if (foundMain)
        {
            //sline 정수 선언
            int sline;

            //tempNode의 typ은 5
            tempNode.type=5;
            //STACK은Push(tempNode,STACK)의 반환값
            STACK=Push(tempNode,STACK);

            //sline은GetLastFunctionCall(STACK)의 반환값
            sline=GetLastFunctionCall(STACK);

            //만약 sline이 0이라면
            if (sline==0)
            {
                // Output = LastExpReturn의값을 콘솔 출력
                printf("Output=%d",LastExpReturn);

            }
            //sline==0 이 아니라면
            else
            {
                //정수 j선언
                int j;
                //정수 foundCall는 0
                int foundCall=0;

                //LastFunctionReturn은 LastExpReturn의 값이다
                LastFunctionReturn=LastExpReturn;
                    //filePtr 파일 닫기
                    fclose(filePtr);
                    /// filePtr= argv[1] 파일 읽기모드는 파일열기
                        filePtr=fopen(argv[1],"r");

                        //curLine은 0
                        curLine=0;
```

```
                        //1부터 sline보다 작을때까지 1씩 증가하며 반복
                        for(j=1;j<sline;j++)
                        {
                            //filePtr의파일을 최대 4095수까지 읽을 수 있으며, dummy배
열에 저장.
                            fgets(dummy,4096,filePtr);
                            //curLine 1증가
                            curLine++;
                        }

                    //만일 foundCall이 0일 경우
                    while(foundCall==0)
                    {
                        //Pop(&tempNode,STACK)실행
                        Pop(&tempNode,STACK);
                        //tempNode의 type이 3일 경우
                        if (tempNode.type==3)
                        {
                            //foundCall은 1이다
                            foundCall=1;
                        }
                    }

            }

        }
    }

    //ⓒ begin과 end가 아니면
    else
    {


        // 공백들을 \0으로 바꿈
        //line에서 공백 전의 첫문자
        firstword=strtok(line," ");


        // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
        //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)

        //firstword가 "int"라면
        if (!strcmpi("int",firstword))
        {
            //foundMain ==0
            //foundMain == 0이면 false 0이 아니면 True
            if (foundMain)
            {
                //ttempNode.type 은 1
                tempNode.type=1; /*integer*/
```

```
            //" "공백 기준으로 다음
            //firstword는 공백으로 쪼개진 문자열
            firstword=strtok(NULL," ");

            //tempNode.exp_data의 exp_data = firstword[0]
            tempNode.exp_data=firstword[0];

            //" "공백 기준으로 다음
            //firstword는 공백으로 쪼개진 문자열
            firstword=strtok(NULL," ");

            // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
            //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
            //firstword가 '='이라면
            if (!strcmpi("=",firstword))
            {
                //" "공백 기준으로 다음
                //firstword는 공백으로 쪼개진 문자열
                firstword=strtok(NULL," ");
            }

            //문자열을 정수 타입 변화 ex) 1
            //tempNode의 val = firstword를 정수 변환
            tempNode.val=atoi(firstword);
            //tempNode의 line은  0
            tempNode.line=0;
            //STACK은 Push(tempNode,STACK)한 값
            STACK=Push(tempNode,STACK);
        }
    }

    // //strcmpi :  둘이 동일하면 0  반환하고 동일하지 않으면 1
    //! 을 붙음으로써 둘이 동일하면 1(True) 동일하지 않으면 0(False)
    // firstword가 "function"이라면
    else if (!strcmpi("function",firstword))
    {
        //tempNode.type는 2이다
        tempNode.type=2;

        //" "공백 기준으로 다음
        //firstword는 공백으로 쪼개진 문자열
        firstword=strtok(NULL," ");

        //tempNode.exp_data는 firstword[0]이다
        tempNode.exp_data=firstword[0];

        //tempNode.line 는 curLine이다
        tempNode.line=curLine;
        //tempNode.val은 0이다.
        tempNode.val=0;
```

```
            //STACK은 Push(tempNode,STACK); 에서의 반환값이다.
            STACK=Push(tempNode,STACK);

            //만약 firstword배열의 0번쨰가 'm'이며 firstword배열의 1번쨰가 'a'이며
firstword배열의 2번쨰가 'i'이며 firstword배열의 3번째까 'n'이라면
            if ( (firstword[0]=='m') & (firstword[1]=='a') & (firstword[2]=='i') &
(firstword[3]=='n') )
            {
                //foundMain 은 1이다.
                foundMain=1;



            }
            //만약 firstword배열의 0번쨰가 'm'이며 firstword배열의 1번쨰가 'a'이며
firstword배열의 2번쨰가 'i'이며 firstword배열의 3번째까 'n'이 아니라면
            else
            {
                //(3)
                //foundMain == 0이면 False 0이 foundMain == 0이 아니면 True
                //foundMain이 0이 아니면 실행
                if (foundMain)
                {

                    //" "공백 기준으로 다음
                    //firstword는 공백으로 쪼개진 문자열
                    firstword=strtok(NULL," ");

                    //tempNode.type 은 1이다
                    tempNode.type=1;

                    //tempNode.exp_data는 firstword[0]이다
                    tempNode.exp_data=firstword[0];

                    //tempNode.val는 CalingFunctionArgVal
                    tempNode.val=CalingFunctionArgVal;

                    //tempNode의line은 0이다;
                    tempNode.line=0;
                    ////STACK은 Push(tempNode,STACK); 에서의 반환값이다.
                    STACK = Push(tempNode, STACK);

                }

            }


        }

        //firstword[0]이 '(' 이라면
        else if (firstword[0]=='(')
        {
```

```
        //foundMain == 0이면 False 0이 foundMain == 0이 아니면 True
        //foundMain이 1이면 실행
        if (foundMain)
            {

            //정수 i는 0
            int i=0;
            //정수 y는 0
            int y=0;

            //MathStack.top은 NULL
            MathStack->top=NULL;




            //lineyedek[i]이 \x0(NULL)'이 아닐때까지 반복
            while(lineyedek[i]!='\x0')
            {
                //lineyedek[i]가 숫자라면 True
                if (isdigit(lineyedek[i])) {
                    //postfix[y에lineyedek[i] 넣기
                    postfix[y]=lineyedek[i];

                    y++;  //y값 1증가

                }

                //lineyedek[i]이 ')'이라면
                else if (lineyedek[i]==')')
                {
                    //0이면 true 1이면 false
                    //0일때
                    if (!isStackEmpty(MathStack) != 0 )
                    {
                        //postfix[y]에 PopOp(MathStack)의 반환값 넣기
                        postfix[y]=PopOp(MathStack);
                        y++; // y값 1증가
                    }
                }

                ////(3)-3-3
                //lineyedek[i]이 '+'이거나 lineyedek[i]이 '-' 이거나 lineyedek[i]이
'+' 이거나 lineyedek[i]이 '/'이라면
                else if ((lineyedek[i]=='+') | (lineyedek[i]=='-') |
(lineyedek[i]=='*') | (lineyedek[i]=='/'))
                {
                    ////(3)-3-3-1
                    /*operators*/
                    //0이면 false 1이면 true
```

```
                    //isStackEmpty(MathStack)의 반환값 0이 아니라면
                    if (isStackEmpty(MathStack) != 0 )
                    {
                        // MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
                        MathStack=PushOp(lineyedek[i],MathStack);
                    }


                    //isStackEmpty(MathStack)의 반환값이 0이라면
                    else
                    {
                        //Priotry(MathStack->top->op)의 반환값이
Priotry(lineyedek[i])의 반환값보다 크거나 같을때
                        if (Priotry(lineyedek[i]) <= Priotry(MathStack->top->op) )
                        {
                            //postfix[y]은 PopOp(MathStack)반환값
                            postfix[y]=PopOp(MathStack);

                            //y값 1증가
                            y++;

                            //MathStack은 PushOp(lineyedek[i],MathStack)이다
                            MathStack=PushOp(lineyedek[i],MathStack);


                        }
                        // Priotry(lineyedek[i])의 반환값이 Priotry(MathStack->top-
>op)의 반환값보다 클 때
                        else
                        {
                            //MathStack은 PushOp(lineyedek[i],MathStack)의 반환값
                            MathStack=PushOp(lineyedek[i],MathStack);
                        }

                    }
                }

                //알파벳 대문자 "A-Z"는 1을 반환.알파벳 소문자 'a-z"는 2를 반환.
                //lineyedek[i]가 영어라면
                else if (isalpha(lineyedek[i])>0)
                {
                    //정수 codeline는 0이다
                    int codeline=0;

                    //정수 dummyint는 0이다
                    int dummyint=0;

                    //정수 retVal는 0이다
                    int retVal=0;
                    //retVal은 GetVal(lineyedek[i],&codeline,STACK);의 반환값
                    retVal=GetVal(lineyedek[i],&codeline,STACK);
```

```
                        //int codeline =1


                        //만일 retVal이 -1이 아니면서 -999이 아니라면
                        if ((retVal!=-1) & (retVal!=-999))
                        {
                            //postfix[y]에retVal+48을한다.
                            postfix[y]=retVal+48;
                            //y에 1증가
                            y++;

                        }
                        ////만일 retVal이 -1이거나 -999이라면
                        else
                        {

                            //만약 LastFunctionReturn은 -999이라면
                            if (LastFunctionReturn==-999)
                            {

                            //정수 j선언
                            int j;

                            //tempNode.type은 3이다.
                            tempNode.type=3;

                            //tempNode.line은 curLine이다
                            tempNode.line=curLine;

                            //STACK은 Push(tempNode,STACK)의 반환값
                            STACK=Push(tempNode,STACK);


                            //CalingFunctionArgVal은
GetVal(lineyedek[i+2],&dummyint,STACK)의 반환값

CalingFunctionArgVal=GetVal(lineyedek[i+2],&dummyint,STACK);


                            //filePtr의 파일을 닫는다.
                            fclose(filePtr);
                            //filePtr은 argv[1]번째를 읽기 모드로 파일을 연다
                            filePtr=fopen(argv[1],"r");
                            //curLine은 0이다.
                            curLine=0;


                            //1부터 codeline보다 작을때까지 1씩 증가하여 반복
                            for(j=1;j<codeline;j++)
                            {
                                //filePtr의파일을 최대 4095수까지 읽을 수 있으며, dummy배
```

```
열에 저장.
                            fgets(dummy,4096,filePtr); /* read the file by Line by
Line */
                            //curLine 1증가
                            curLine++;
                        }

                        //WillBreak은 1이다.
                        WillBreak=1;

                        //whlie문 나가기
                        break;
                        }


                        //만약 LastFunctionReturn은 -999아니라면
                        else

                        {

                        //postfix[y]은LastFunctionReturn+48이다.
                        postfix[y]=LastFunctionReturn+48;
                        y++; // y값 1증가
                        i=i+3; //i는 i+3이다.
                        LastFunctionReturn=-999; //LastFunctionReturn은 -999이다.


                        }
                    }
                    }
                }

                //i에 1 증가
                i++;
            }//while문



            //WillBreak이 0이라면
            if (WillBreak==0)
            {

            //isStackEmpty(MathStack)이 0이라면 반복
            while (isStackEmpty(MathStack)==0)
            {

                //postfix[y]= PopOp(MathStack)의 반환값
                postfix[y]=PopOp(MathStack);
                //y값 1증가
                y++;
            }
```

```
            //postfix[y]에 '\0'을 넣는다.
            postfix[y]='\0';


            // i 는 0
            i=0;

            //CalcStack의 top은 NULL
            CalcStack->top=NULL;

            //postfix[i]이 \x0일때까지 반복
            while(postfix[i]!='\x0')
            {
                //postfix[i]가 숫자라면
                if (isdigit(postfix[i])) {

                    //CalcStack PushPostfix(postfix[i]-'0',CalcStack);의 반환값
                    CalcStack=PushPostfix(postfix[i]-'0',CalcStack);

                }
                //postfix[i]가 '+'이거나 postfix[i]가 '-' 이거나 postfix[i]가 '*'
이거나 postfix[i]가 '/' 이라면
                else if ((postfix[i]=='+') | (postfix[i]=='-') | (postfix[i]=='*')
| (postfix[i]=='/'))
                {
                    //val1 은 PopPostfix(CalcStack)의 반환값
                    val1=PopPostfix(CalcStack);

                    //val2 은 PopPostfix(CalcStack)의 반환값
                    val2=PopPostfix(CalcStack);

                    //switch은
                    //postfix[i]은 판단할 값으로 case와 입력값이 있다면 해당 case문
실행
                    switch (postfix[i])
                    {
                        //postfix[i]가 '+'일때 resultVal는 val2+val1의 값이다.
                        //후 break문으로 switch문 빠져나간다.
                        case '+': resultVal=val2+val1;break;
                        //postfix[i]가 '-' 일때 resultVal는 val2-val1의 값이다.
                        //후 break문으로 switch문 빠져나간다
                        case '-': resultVal=val2-val1;break;

                        ///postfix[i]가 '/'일때 resultVal는 val2/val1의 값이다
                        //후 break문으로 switch문 빠져나간다
                        case '/': resultVal=val2/val1;break;

                        //postfix[i]가 '*'일때 resultVal는 val2*val1의 값이다.
                        // //후 break문으로 switch문 빠져나간다
                        case '*': resultVal=val2*val1;break;
                    }
```

```
                    //CalcStack 은 PushPostfix(resultVal,CalcStack)의 반환값
                    CalcStack=PushPostfix(resultVal,CalcStack);
                }
                // i에 1 증가
                i++;
            }


            //LastExpReturn 은 CalcStack의  top의 val이다.
            LastExpReturn=CalcStack->top->val;


            }

            //WillBreak 는 0
            WillBreak=0;
            }
        }
    }


}
//filePtr 파일 닫기
fclose(filePtr);
//printAllStack(STACK);

//STACK은FreeAll(STACK)의 반환값
STACK=FreeAll(STACK);

//Press a key to exit...콘솔에 출력
printf("\nPress a key to exit...");
//키를 입력받는다.(input)
getch();

//반환값 0
return  0;
```

}