Hw2 Parentheses Checker

데이터에 좌우 괄호가 있을때, 괄호의 균형을 맞추었는지 스택을 사용하여 테스트 할 수 있는 프로그램을 작성하시오 (Write a program to read a text file and print whether or not the parentheses are **balanced** in the expression. (use **stack**))

1) 다음 데이터들로 데이터 파일(hw2.txt)을 구성 하시오.

(The *data file (*hw2.txt) should contain the following data)

1. (A + B) - { C + D } - [F + G]
2. (A * (B + (C * D + E)))
3. ((A + B)
4. (A + B)) - (C + D
5. (A+B)}
6. (A + B })
7. (A + B) - (C + D]}
8. { [A + B) - [(C -D)] }

2) Output

Input Data 1. (A + B) - { C + D } - [F + G]
The Expression is Valid

Input Data 2. (A * (B + (C * D + E)))
The Expression is Valid

Input Data 3. ((A + B)
The Expression has unbalanced parentheses
The Expression is Invalid

Input Data 4. (A + B)) - (C + D
The Expression has unbalanced parentheses
The Expression is Invalid

Input Data 5. (A+B)}
The Expression has unbalanced parentheses
The Expression is Invalid

Input Data 6. (A + B })
The Mismatched Parenthes in the Expression are (and }
The Expression is Invalid

Input Data 7. (A + B) - (C + D]}
The Mismatched Parenthes in the Expression are (and]
The Expression is Invalid

Input Data 8. { [A + B) - [(C -D)] }
The Mismatched Parenthes in the Expression are [and)
The Expression is Invalid

Total: Balanced :2 Unbalanced :3 Mimatched : 3

- Invalid 의 경우 아래 두가지로 구분할 것
  - Parentheses 의 개수가 틀린 경우 (위 예제 3, 4, 5)
  - Parentheses 의 종류가 틀린 경우 (위 예제 6, 7, 8)
    종류가 틀린 경우 “틀린 parentheses”를 명시할 것.

Algorithm 참조:

```
procedure main() {
    open data file // check file open error
    while (infile.getline(buffer, 80)) {    // check line by line (한 라인씩 검사)
        validity = check_paren (buffer );
        if (validity is true) print “valid”      else print “Invalid”
    }
     Print Total Number of Each Parentheses;    // balanced, unbalanced, mismatched
 }

int check_paren(exp) {
  for(i=0; i < strlen(exp); i++){
       if(exp[i]=='(' || exp[i]=='{' || exp[i]=='[')      push(exp[i]);
       if(exp[i]==')' || exp[i]=='}' || exp[i]==']')    {
            if (stack empty) {    print("UnBalanced "); Unbalanced++; return}
            else { temp=pop();
                    if(!match(temp, exp[i]))    {
                       print("Mismatched”    temp, “and “ exp[i]);
                       mismatched++; return}
                    } }
          }// end of if
  } // end of for loop

  if(stack empty)    return true
  else    { unbalanced++;      return false;    }      // stack 에  parentheses
  }

int match(a, b){
     if (a= ‘[‘ and b= ‘]’) return true;
     else if (a =’{‘ and b = ‘}’ return true;
     else if (a= ‘(‘ and b= ‘)’ return true
     return 0}

void push(char data);char pop();void printStack();void createStack();
```

- 실행화면

```
<< Hw2:  Balanced Parentheses >>

Input Data 1. ( A + B ) - { C + D } - [F + G]
The Expression is Valid

Input Data 2. (A * ( B + ( C * D + E)))
The Expression is Valid

Input Data 3. ( ( A + B)
The Expression has unbalanced parentheses
The Expression is Invalid

Input Data 4. (A + B)) - (C + D
The Expression has unbalanced parentheses
The Expression is Invalid

Input Data 5. (A+B)}
The Expression has unbalanced parentheses
The Expression is Invalid

Input Data 6. (A + B } )
 The Mismatched Parenthes in the Expression are ( and }
The Expression is Invalid

Input Data 7. (A + B) - (C + D]}
 The Mismatched Parenthes in the Expression are ( and ]
The Expression is Invalid

Input Data 8. { [ A + B ) - [ ( C -D ) ] }
 The Mismatched Parenthes in the Expression are [ and )
The Expression is Invalid

 Total:  Balanced :2 Unbalanced :3 Mimatched : 3
```