Homework #4 B09705039 劉惟恩

Solutions:

1.

(1) ***Pseudocode function:***

```
bool balancedBracketProblem(string a)
{
   //step1 : check every character in the string and solve the question by stack ADT
   for (each character ch in the string)
   {
      if (ch is a left bracket(e.g."(", "[", "{"))
         aStack.push(char);
      else if (ch is a right bracket(e.g.")", "[", "{"))
      {
         if (aStack.peek() == the same type of its left bracket(e.g."(", "[", "{"))
            aStack.pop();
         else    // the left bracket doesn't match the right bracket ==> invalid
         {
            cout << "The string is Invalid !!!";
            return false;
         }
      }
   }

   //step2 : check if there are any left brackets left in the stack
   if(aStack.isEmpty() == true)
   {
      cout << "The string is Valid !!!";
      return true;
   }
   else    // still exists at least one left bracket in the stack ==> invalid
   {
      cout << "The string is Invalid !!!";
      return false;
   }
}
```

(2) ***Simulate:***

stack ADT will be implemented as: ***Stack [a (bottom), b, c (top)

(a)

start : **aStack [**

step1 : read in “ [” ==> **aStack [“ [”**
step2 : read in “ (” ==> **aStack [“ [” , “ (”**
step3 : read in “) ” , pop “ (” out ==> **aStack [“ [”**
step4 : read in “ { ” ==> **aStack [“ [” , “ { ”**
step5 : read in “ } ” , pop “ { ” out ==> **aStack [“ [”**

step6 : read in “ (” ==> **aStack [“ [” , “ (”**
step7 : read in “) ” , pop “ (” out ==> **aStack [“ [”**
step8 : read in “] ” , pop “ [” ==> **aStack [**

end : **aStack [**
result : aStack is empty at the end so it is valid, ***cout << "The string is Valid !!!”*** and ***return true***

(b)
start : **aStack [**

step1 : read in “ (” ==> **aStack [“ (”**
step2 : read in “ (” ==> **aStack [“ (” , “ (”**
step3 : read in “) ” , pop “ (” out ==> **aStack [“ (“**
step4 : read in “ { ” ==> **aStack [“ (” , “ { ”**
step5 : read in “ (” ==> **aStack [“ (” , “ { ” , “ (”**
step6 : read in “ [” ==> **aStack [“ (” , “ { ” , “ (” , ” [”**
step7 : read in “] ” , pop “ [” out ==> **aStack [“ (” , “ { ” , “ (”**
step8 : read in “) ” , pop “ (” out ==> **aStack [“ (” , “ { ”**
step9 : read in “ } ” , pop “ { ” out ==> **aStack [“ (”**
step9 : read in “) ” , pop “ (” out ==> **aStack [**

end : **aStack [**
result : aStack is empty at the end so it is valid, ***cout << "The string is Valid !!!”*** and ***return true***

2.
(a)
start : **aStack [** ; postfixExp

step1 : read in “a”, append to postfixExp
step2 : read in “+” ==> **aStack [“+"**
step3 : read in “b”, append to postfixExp
step4 : read in “-”, append “+” to postfixExp, pop “+” out ==> **aStack [“-"**
step5 : read in “c”, append to postfixExp
step6 : append “-” to postfixExp, pop “-” out ==> **aStack [**

end : **aStack [**
result : postfixExp = ab+c-

(b)
start : **aStack [**; postfixExp

step1 : read in “(” ==> **aStack [“(”**
step2 : read in “a”, append to postfixExp
step3 : read in “+” ==> **aStack [“(” , “+”**
step4 : read in “b”, append to postfixExp
step5 : read in “)”, append “+” to postfixExp, pop “+” and “(” out ==> **aStack [**
step6 : read in “*” ==> **aStack [“*”**
step7 : read in “(” ==> **aStack [“*” , “(”**
step8 : read in “c”, append to postfixExp
step9 : read in “-” ==> **aStack [“*” , “(” , “-”**

step10 : read in “d”, append to postfixExp
step11 : read in “)” , append “-” to postfixExp, pop “-” and “(” out ==> **aStack [“*”**
step12 : append “*” to postfixExp, pop “*” out ==> **aStack [**

end : **aStack [**
result : postfixExp = ab+cd-*

(c)
start : **aStack [**; postfixExp

step1 : read in “(” ==> **aStack [“(”**
step2 : read in “a”, append to postfixExp
step3 : read in “*” ==> **aStack [“(” , “*”**
step4 : read in “(” ==> **aStack [“(” , “*” , “(”**
step5 : read in “b”, append to postfixExp
step6 : read in “*” ==> **aStack [“(” , “*” , “(” , “*”**
step7 : read in “c”, append to postfixExp
step8 : read in “)”, append “*” to postfixExp, pop “*” and “(” out ==> **aStack [“(” , “*”**
step9 : read in “)”, append “*” to postfixExp, pop “*” and “(” out ==> **aStack [**
step10 : read in “-” ==> **aStack [“-”**
step11 : read in “d”, append to postfixExp
step12 : read in “+” , append “-” to postfix, pop “-” out ==> **aStack [“+”**
step13 : read in “e”, append to postfixExp
step14 : read in “/” ==> **aStack [“+” , “/”**
step15 : read in “f”, append to postfixExp
step16 : append “/” to postfixExp, pop “/” out ==> **aStack [“+”**
step17 : append “+” to postfixExp, pop “+” out ==> **aStack [**

end : **aStack [**
result : postfixExp = abc**d-ef/+

(d)
start : **aStack [**; postfixExp

step1 : read in “a”, append to postfixExp
step2 : read in “/” ==> **aStack [“/”**
step3 : read in “(” ==> **aStack [“/” , “(”**
step5 : read in “b”, append to postfixExp
step6 : read in “-” ==> **aStack [“/” , “(” , “-”**
step7 : read in “c”, append to postfixExp
step8 : read in “)”, append “-” to postfixExp, pop “-” and “(” out ==> **aStack [“/”**
step9 : read in “+” , append “/” to postfixExp, pop “/” out ==> **aStack [“+”**
step10 : read in “(” ==> **aStack [“+” , “(”**
step11 : read in “d”, append to postfixExp
step12 : read in “+” ==> **aStack [“+” , “(” , “+”**
step13 : read in “e”, append to postfixExp
step14 : read in “)”, append “+” to postfixExp, pop “+” and “(” out ==> **aStack [“+”**
step15 : read in “*” ==> **aStack [“+” , “*”**
step16 : read in “f”, append to postfixExp
step17 : append “*” to postfixExp, pop “*” out ==> **aStack [“+”**
step18 : append “+” to postfixExp, pop “+” out ==> **aStack [**

end : **aStack [**
result : postfixExp = abc-/de+f*+

(e)
start : **aStack [**; postfixExp

step1 : read in “(” ==> **aStack [“(”**
step2 : read in “(” ==> **aStack [“(” , “(”**
step3 : read in “a”, append to postfixExp
step4 : read in “+” ==> **aStack [“(” , “(” , “+”**
step5 : read in “b”, append to postfixExp
step6 : read in “)”, append “+” to postfixExp, pop “+” and “(” out ==> **aStack [“(”**
step7 : read in “*” ==> **aStack [“(” , ”*”**
step8 : read in “c”, append to postfixExp
step9 : read in “-”, append “*” to postfixExp, pop “*” out ==> **aStack [“(” , ”-”**
step10 : read in “(” ==> **aStack [“(” , ”-” , “(”**
step11 : read in “d”, append to postfixExp
step12 : read in “-” ==> **aStack [“(” , ”-” , “(” , “-”**
step13 : read in “e”, append to postfixExp
step14 : read in “)”, append “-” to postfixExp, pop “-” and “(” out ==> **aStack [“(” , ”-”**
step15 : read in “)”, append “-” to postfixExp, pop “-” and “(” out ==> **aStack [**
step16 : read in “*” ==> **aStack [“*”**
step17 : read in “(” ==> **aStack [“*” , “(”**
step18 : read in “f”, append to postfixExp
step19 : read in “+” ==> **aStack [“*” , “(” , ”+”**
step20 : read in “g”, append to postfixExp
step21 : read in “)”, append “+” to postfixExp, pop “+” and “(” out ==> **aStack [“*”**
step22 : append “*” to postfixExp, pop “*” out ==> **aStack [**

end : **aStack [**
result : postfixExp = ab+c*de- -fg+*

(f)
start : **aStack [**; postfixExp

step1 : read in “a”, append to postfixExp
step2 : read in “+” ==> **aStack [“+”**
step3 : read in “(” ==> **aStack [“+” , “(”**
step4 : read in “b”, append to postfixExp
step5 : read in “*” ==> **aStack [“+” , “(” , “*”**
step6 : read in “c”, append to postfixExp
step7 : read in “/”, append “*” to postfixExp, pop “*” out ==> **aStack [“+” , “(” , “/”**
step8 : read in “d”, append to postfixExp
step9 : read in “)”, append “/” to postfixExp, pop “/” and “(” out ==> **aStack [“+”**
step10 : read in “-”, append “+” to postfixExp, pop “+” out ==> **aStack [”-”**
step11 : read in “e”, append to postfixExp
step12 : append “-” to postfixExp, pop “-” out ==> **aStack [**

end : **aStack [**
result : postfixExp = abc*d/+e-