1.

```
checkBraces(aString: string): boolean

aStack = a new empty stack

for (int index = 0; index < aString.length(); index++)
{
      ch = character at position index in aString

      if (ch is ‘(’ or  ‘[’ or  ‘{‘)
      {
            aStack.push(ch)
            continue
      }

      switch (ch)
      {
            case ')':
                  if (aStack.peek() == '{' || x == '[')
                      return false;
                  aStack.pop()
                  break;

             case '}':
                  if (aStack.peek() == '(' || x == '[')
                      return false;
                  aStack.pop()
                  break;

             case ']':
                  if (aStack.peek() == '(' || x == '{')
                      return false;
                  aStack.pop()
                  break;
      }
}

// Check Empty Stack
return (s.empty());
```

a. $[(a + b) - \{c + d\} + (x + y)]$

		aStack
Step1	push('[')	[
Step2	push('(')	[(
Step3	pop( )	[
Step4	push('{')	[{
Step5	pop( )	[
Step6	push('(')	[(
Step7	pop( )	[
Step8	pop( )	

b. ((a) * {([b + c])})

		aStack
Step1	push('(')	(
Step2	push('(')	((
Step3	pop( )	(
Step4	push('{')	({
Step5	push('(')	({ (
Step6	push('[')	({ ([
Step7	pop( )	({ (
Step8	pop( )	({
Step9	pop( )	(
Step10	pop( )	

a. a + b - c

Char	Stack	PostfixExp	Others
a		a	
+	+	a	
b	+	ab	
-	-	ab+	
c	-	ab+c	
		ab+c-	Copy operators from stack to postfixExp

b. (a + b) * (c - d)

Char	Stack	PostfixExp	Others
(	(		
a	(	a	
+	(+	a	
b	(+	ab	
)	(	ab+ ab+	Move operators from stack to postfixExp until "("
*	*	ab+	
(	*(	ab+	
c	*(	ab+c	
-	*(-	ab+c	
d	*(-	ab+cd	
)	*(*	ab+cd- ab+cd- ab+cd-*	Move operators from stack to postfixExp until "(" Copy operators from stack to postfixExp

c. (a * (b * c)) - d + e / f

Char	Stack	PostfixExp	Others
(	(		
a	(	a	
*	(*	a	
(	(*(	a	

Char	Stack	PostfixExp	Others
b	(*(	ab	
*	(*(*	ab	
c	(*(*	abc	
)	(*((*	abc* abc*	Move operators from stack to postfixExp until "("
)	(	abc** abc**	Move operators from stack to postfixExp until "("
-	-	abc**	
d	-	abc**d	
+	-+	abc**d	
e	-+	abc**de	
/	-+/	abc**de	
f	-+/	abc**def abc**def/+-	Copy operators from stack to postfixExp

d. a / (b - c) + (d + e) * f

Char	Stack	PostfixExp	Others
a		a	
/	/	a	
(	/(	a	
b	/(	ab	
-	/(-	ab	
c	/(-	abc	
)	/(/	abc- abc-	Move operators from stack to postfixExp until "("
+	/+	abc-	
(	/+(	abc-	
d	/+(	abc-d	
+	/+(+	abc-d	
e	/+(+	abc-de	
)	/+(/+	abc-de+ abc-de+	Move operators from stack to postfixExp until "("

Char	Stack	PostfixExp	Others
*	/+*	abc-de+	
f	/+*	abc-de+f abc-de+f*+/	Copy operators from stack to postfixExp

e. ((a + b) * c - (d - e)) * (f + g)

Char	Stack	PostfixExp	Others
(	(		
(	((		
a	((	a	
+	((+	a	
b	((+	ab	
)	(((	ab+ ab+	Move operators from stack to postfixExp until "("
*	(*	ab+	
c	(*	ab+c	
-	(-	ab+c*	
(	(-(	ab+c*	
d	(-(	ab+c*d	
-	(-(-	ab+c*d	
e	(-(-	ab+c*de	
)	(-((-	ab+c*de- ab+c*de-	Move operators from stack to postfixExp until "("
)	(	ab+c*de-- ab+c*de--	Move operators from stack to postfixExp until "("
*	*	ab+c*de--	
(	*(	ab+c*de--	
f	*(	ab+c*de--f	
+	*(+	ab+c*de--f	
g	*(+	ab+c*de--fg	
)	*(*	ab+c*de--fg+ ab+c*de--fg+ ab+c*de--fg+*	Move operators from stack to postfixExp until "(" Copy operators from stack to postfixExp

f. a + (b * c / d) - e

Char	Stack	PostfixExp	Others
a		a	
+	+	a	
(	+(	a	
b	+(	ab	
*	+(*	ab	
c	+(*	abc	
/	+(*/	abc	
d	+(*/	abcd	
)	+(* +(+	abcd/ abcd/* abcd/*	Move operators from stack to postfixExp until "("
-	+-	abcd/*	
e	+-	abcd/*e abcd/*e+-	Copy operators from stack to postfixExp