Braces

CS110C

Max Luttrell, CCSF

problem: balanced braces

- C++ uses curly braces to delimit groups of statements
- Let's treat a C++ program as one long string. How can we determine if its braces are balanced?
  - Example: {abc{de}fg{hij}k} is balanced
  - Example: {abc{de}fg{hijk} is not balanced
- Conditions:
  - Each } must match a previously-seen {
  - When you get to the end, all {'s have been matched

ADT stack use balanced braces

- Here's a start...

```
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
    aStack.pop()
}
```

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{**a**{b}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{**b**}c}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}**c**}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b}c}

string

aStack

true

balancedSoFar

ADT stack use
balanced brac

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.p
  }
}

if (balancedSoFa                ck.isEmpty())
  string is bala
else
  string is not balanced
```

{a{b}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{**a**{bc}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{**b**c}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{b**c**}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

{

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = fals
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a{bc}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{**a**b}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{a**b**}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

{

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}**c**}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

aStack

true

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = false
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

aStack

false

balancedSoFar

ADT stack use balanced braces

```
boolean balancedSoFar = true
for (each character c in the string)
{
  if (c is a '{')
    aStack.push('{')
  else if (c is a '}')
  {
    if aStack.isEmpty()
      balancedSoFar = fals
    else
      aStack.pop()
  }
}

if (balancedSoFar and aStack.isEmpty())
  string is balanced
else
  string is not balanced
```

{ab}c}

string

aStack

false

balancedSoFar