Pseudo Code

What is Pseudo Code

- High level and informal set of instructions used to describe an algorithm
- NOT a programming language
- Used to plan programs (Usually the next step after flow chart design)
- No standards
  - Everyone does it their own way

Pseudo Code: Variable Expression

```
<VARIABLE NAME> = <EXPRESSION>
```

Pseudo Code: Variable Expression

```
x = 5
y = 10
z = x + y
a = (z + x) * (z + y)
```

Pseudo Code: Function Expression

```
Function <FUNCTION NAME> (<ARGUMENTS>)
{
  <INSTRUCTIONS>
  Return <VALUE>
}
```

Pseudo Code: Function Expression

```
Function Power (X, Y)
{
  C = 0
  A = 1
  < DO POWER PROCESS >
  Return A
}
```

Pseudo Code: Call Function

```
<FUNCTION NAME> ( <ARGUMENTS> )
```

Pseudo Code: Call Function

```
BASE = 3
EXPONENT = 4
ANSWER = Power(BASE, EXPONENT)
```

Pseudo Code: Decision Statement

- If/Else Decision
  - A decision rendered by Boolean expressions that follow a logical flow of conditions
  - IF something is true, do something

Pseudo Code: Decision Statement

```
If (<CONDITION>)
{
     <INSTRUCTIONS>
}
```

```
If (<CONDITION>)
{
     <INSTRUCTIONS>
}
Else
{
     <INSTRUCTIONS>
}
```

```
If (<CONDITION>)
{
     <INSTRUCTIONS>
}
Else If (<CONDITION>)
{
     <INSTRUCTIONS>
}
Else
{
     <INSTRUCTIONS>
}
```

Pseudo Code: Decision Statement

```
If (<CONDITION> && <CONDITION>)
{
     <INSTRUCTIONS>
}
```

```
If (<CONDITION> || <CONDITION>)
{
     <INSTRUCTIONS>
}
```

```
If (!<CONDITION>)
{
     <INSTRUCTIONS>
}
```

```
If (<CONDITION> || <CONDITION> && <CONDITION>)
{
     <INSTRUCTIONS>
}
```

```
If (<CONDITION>)
{
     If (<CONDITION>)
     {
          <INSTRUCTIONS>
     }
}
```

```
If (<CONDITION> || (<CONDITION> && <CONDITION>))
{
     <INSTRUCTIONS>
}
```

Pseudo Code: Decision Statement

- `<CONDITION>` must be a YES or NO expression
  - Example: If (< **Does X = Y?** >)

Pseudo Code: Decision Statement

```
BASE = 3
EXPONENT = 4
ANSWER = Power(BASE, EXPONENT)
If (ANSWER > 10)
{
   PRINT “ANSWER is greater then 10”
}
Else If (ANSWER >= 5 && ANSWER <= 9)
{
   PRINT “ANSWER is between 5 and 9”
}
Else
{
   PRINT “ANSWER is less then 5”
}
```

Pseudo Code: Looping (While)

```
While (<CONDITION>)
{
  <INSTRUCTIONS>
}
```

Pseudo Code: Looping (While)

```
Function Power (X, Y)
{
  C = 0
  A = 1
  While (C < Y)
  {
            C = C + 1
            A = A * X
  }
  Return A
}
```

Pseudo Code: Looping (Do-While)

```
Do
{
  <INSTRUCTIONS>
}
While (<CONDITION>)
```

Pseudo Code: Looping (Do-While)

```
X = 1
Y = 10
Do
{
  PRINT X
   X = X + 1
}
While (X <= Y)
```

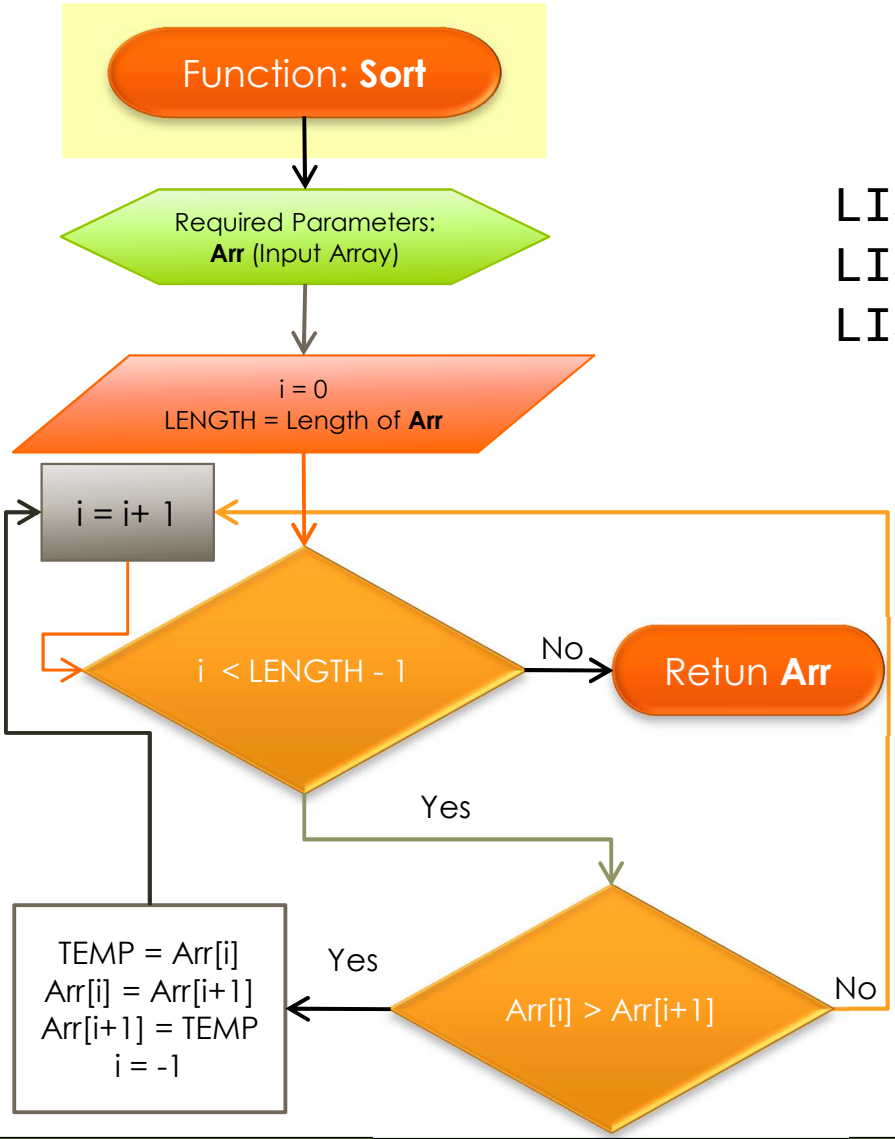
Pseudo Code: Looping (For)

```
For (<START EXPRESSION>;<CONDITION>;<ITERATION EXPRESSION>)
{
   <INSTRUCTIONS>
}
```

Pseudo Code: Looping (For)

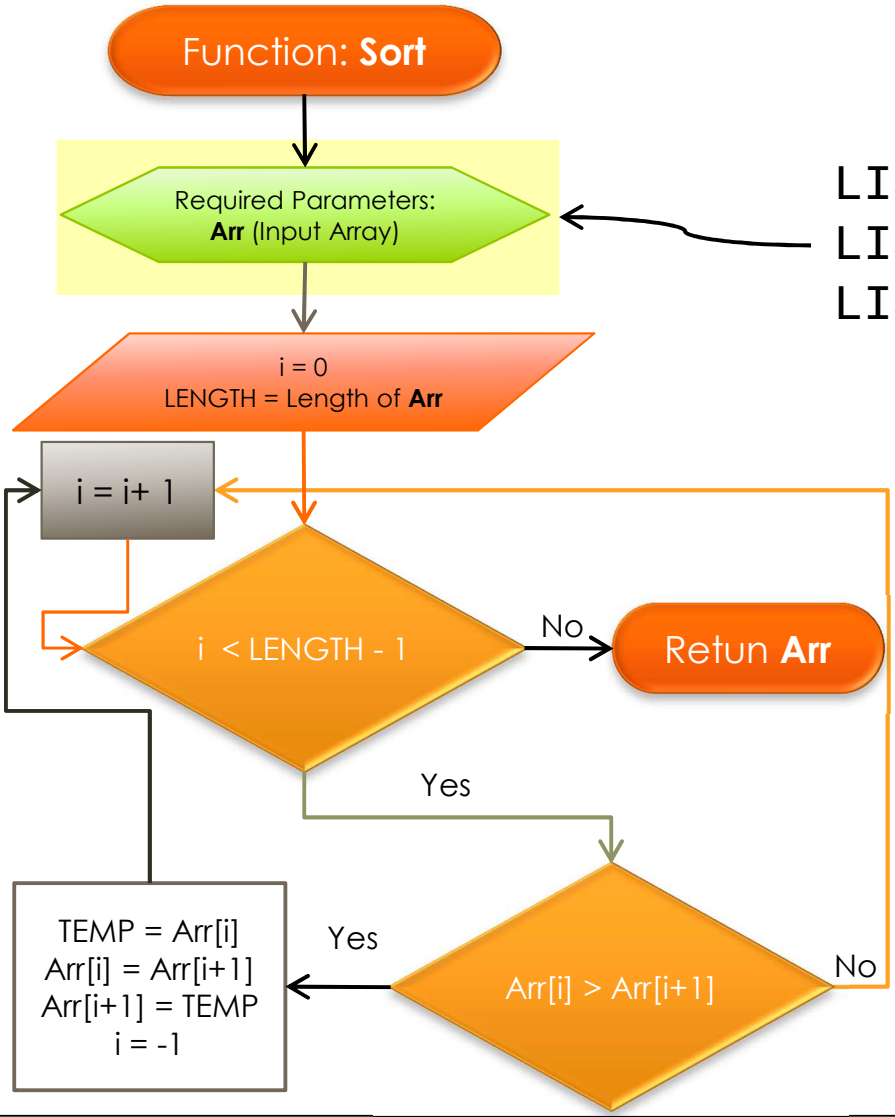
```
Y = 10
For (X = 1; X <= Y; X++)
{
  PRINT X
}
```

Pseudo Code



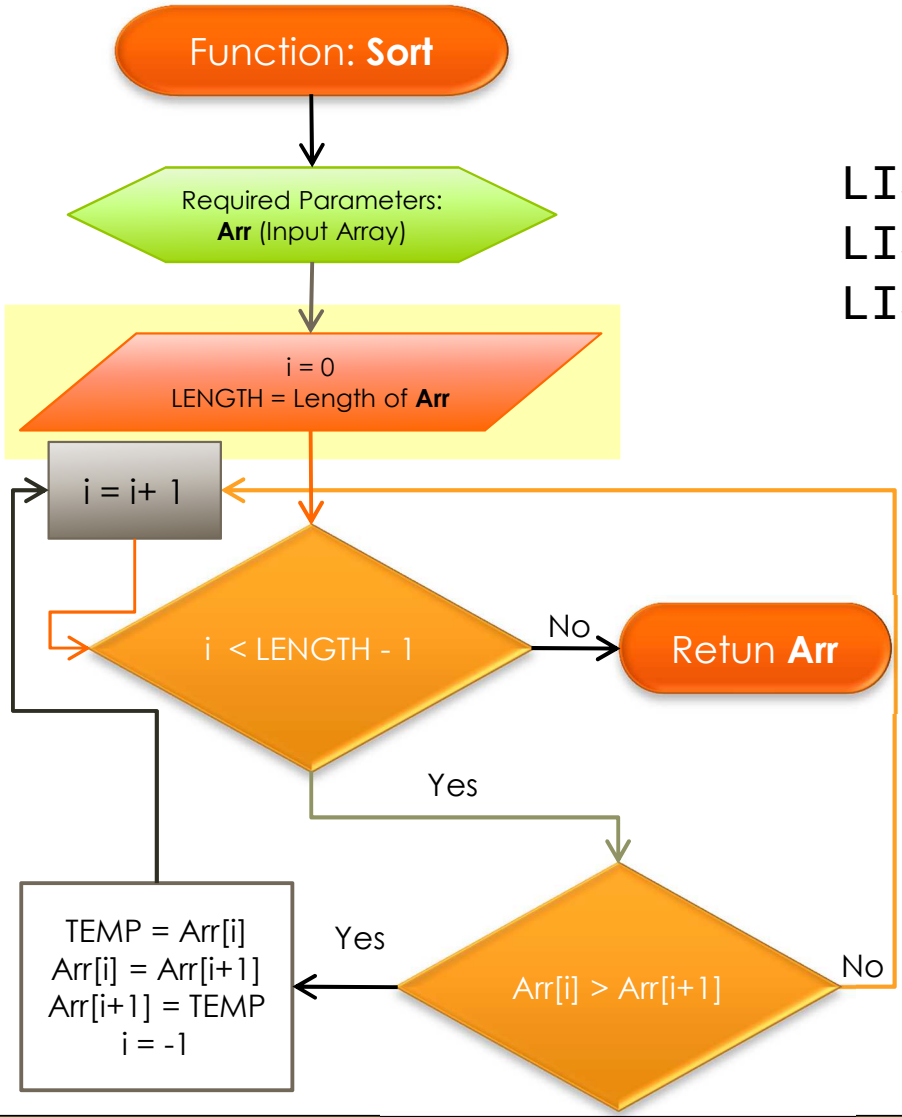

```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

Pseudo Code




```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

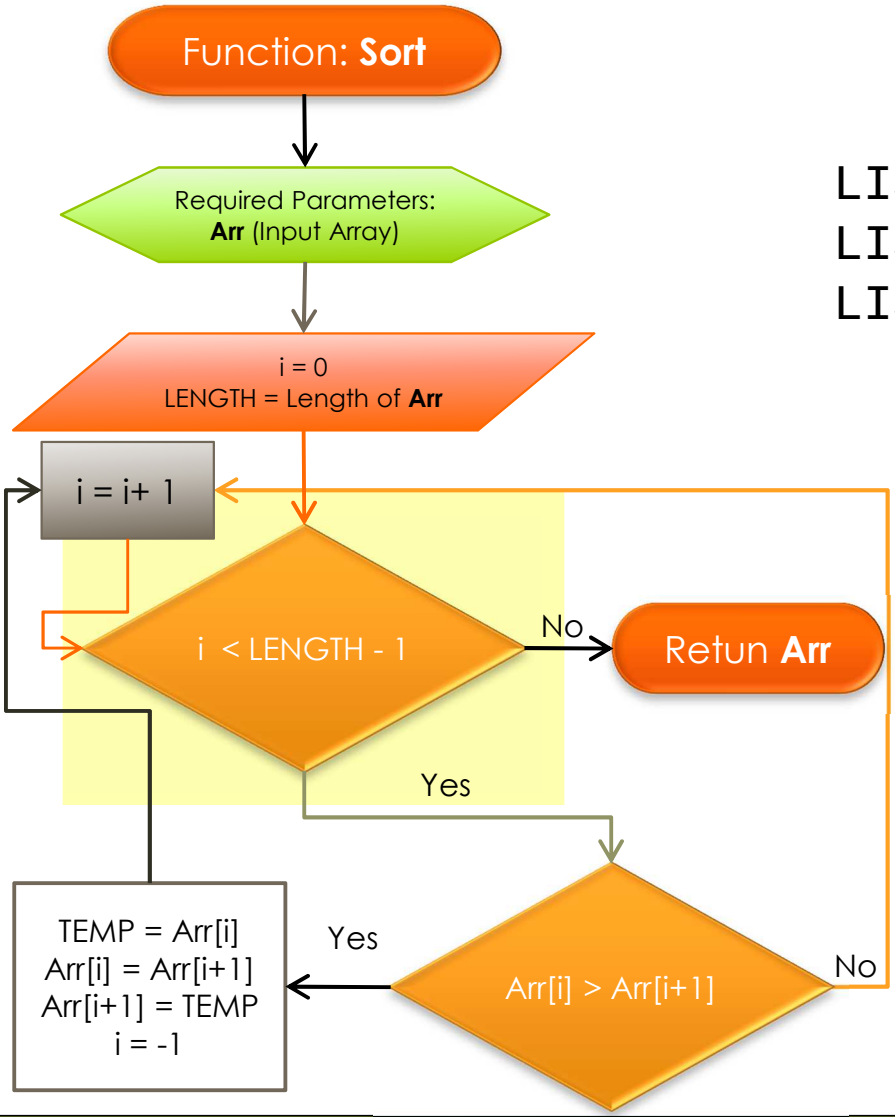
Pseudo Code




```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

```
i = 0
LENGTH = 3
```

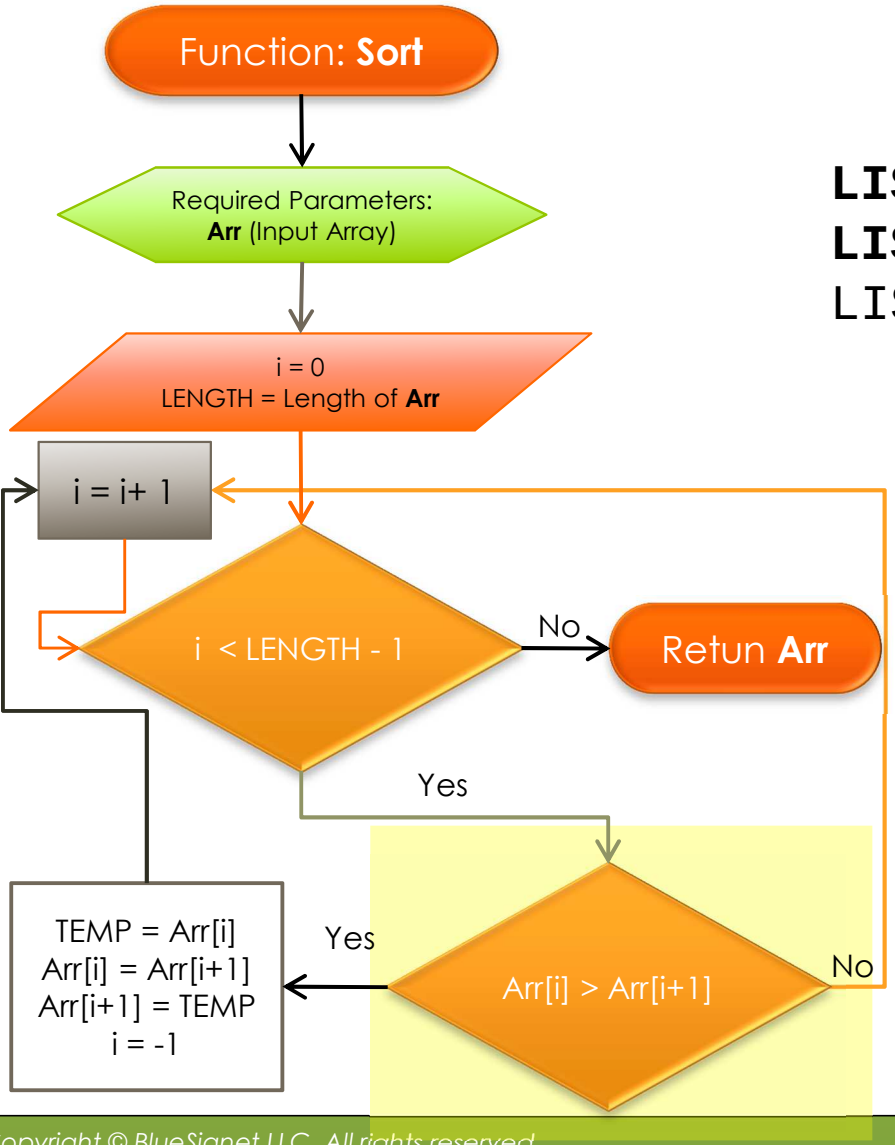
Pseudo Code




```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

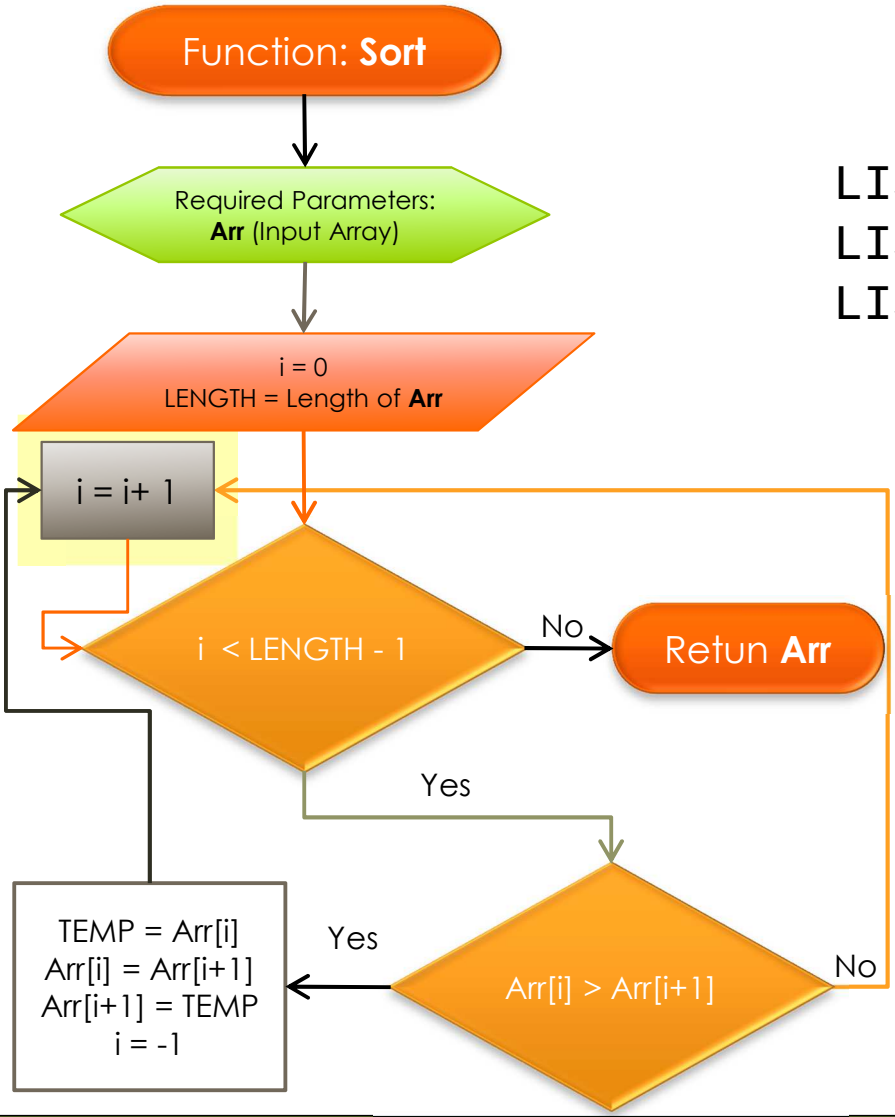
```
i = 0
LENGTH = 3
(0 < 2) = YES
```

Pseudo Code




```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

```
i = 0
LENGTH = 3
(1 > 2) = NO
```

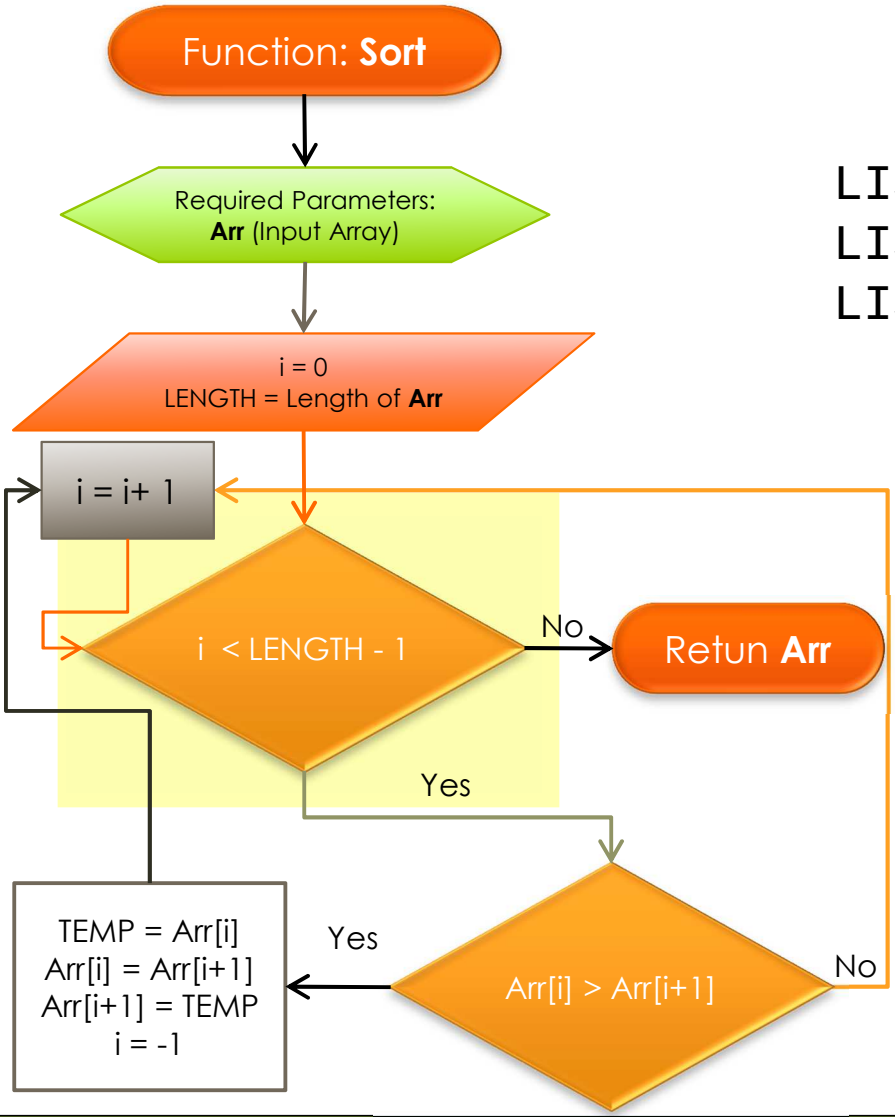


Pseudo Code

```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

```
i = 1
LENGTH = 3
```

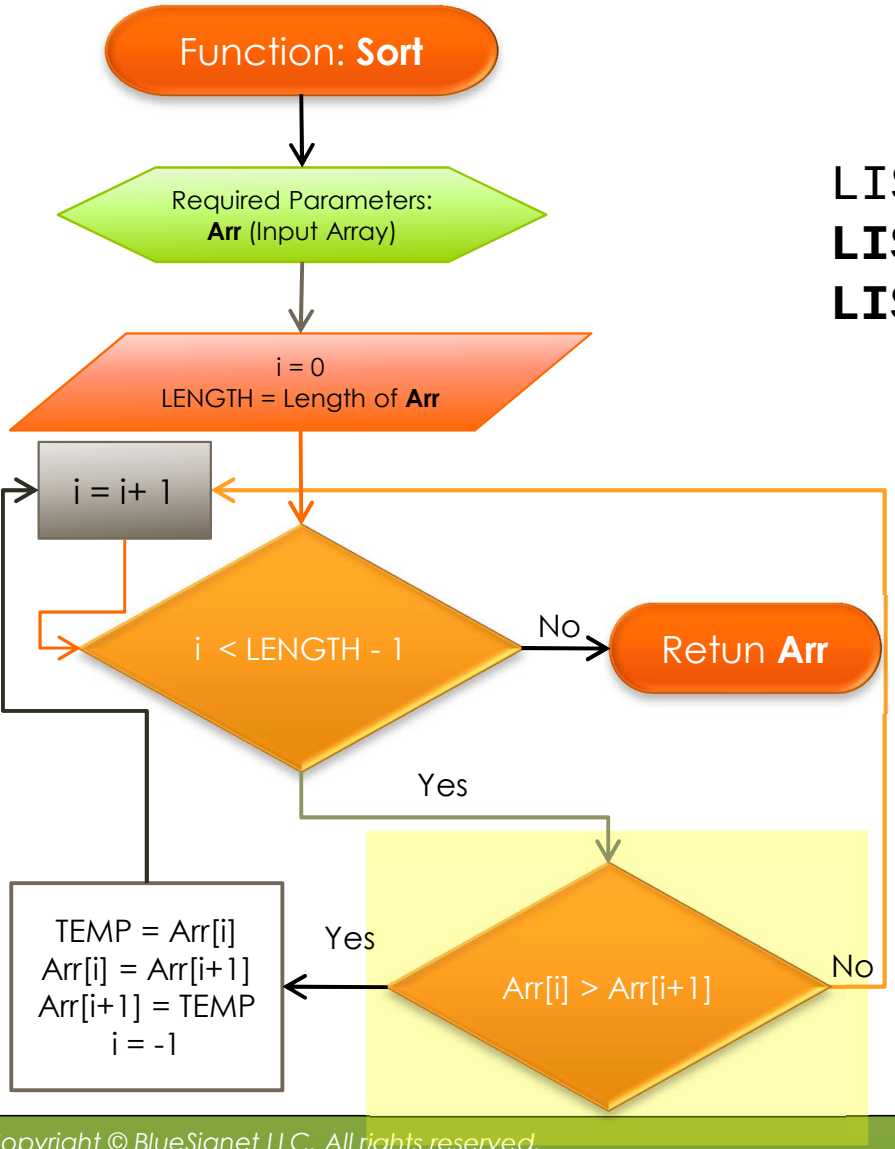
Pseudo Code




```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

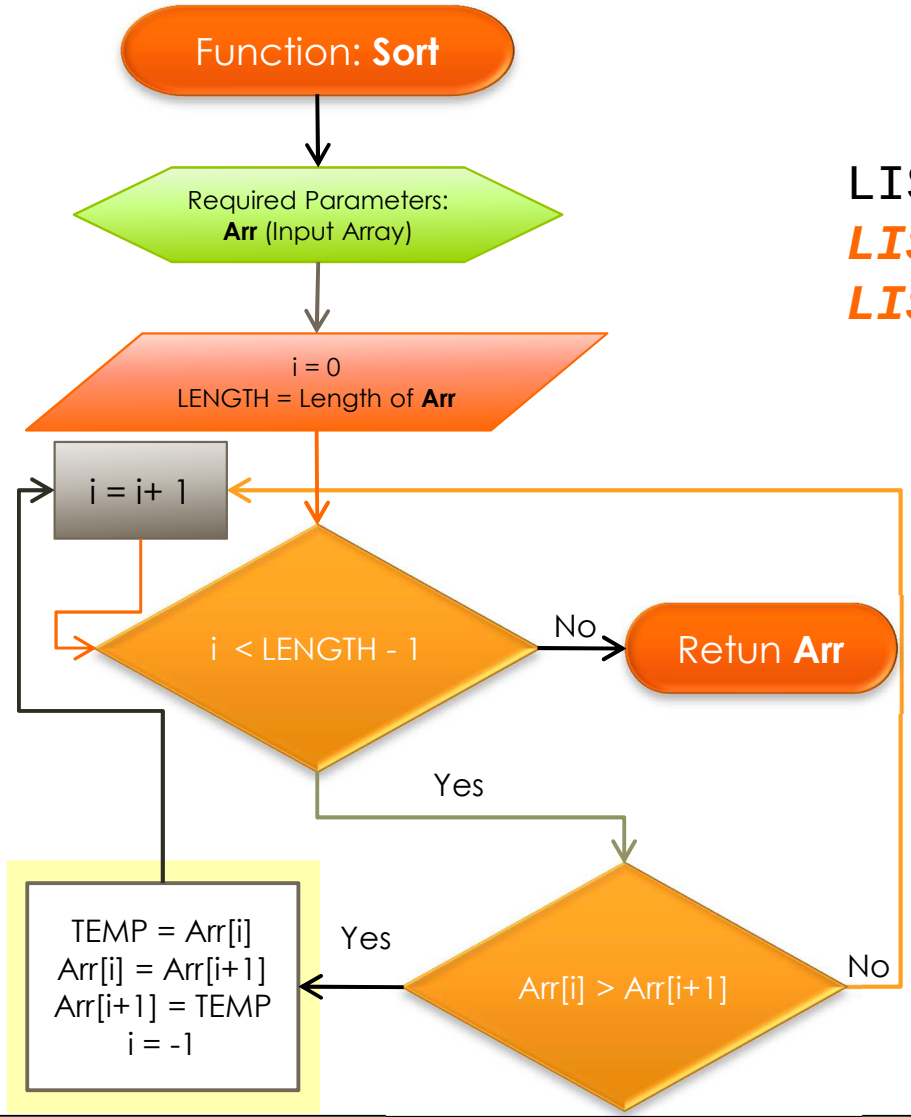
```
i = 1
LENGTH = 3
(1 < 2) = YES
```

Pseudo Code




```
LIST[0] = 1
LIST[1] = 2
LIST[2] = 0
```

```
i = 1
LENGTH = 3
(2 > 0) = YES
```

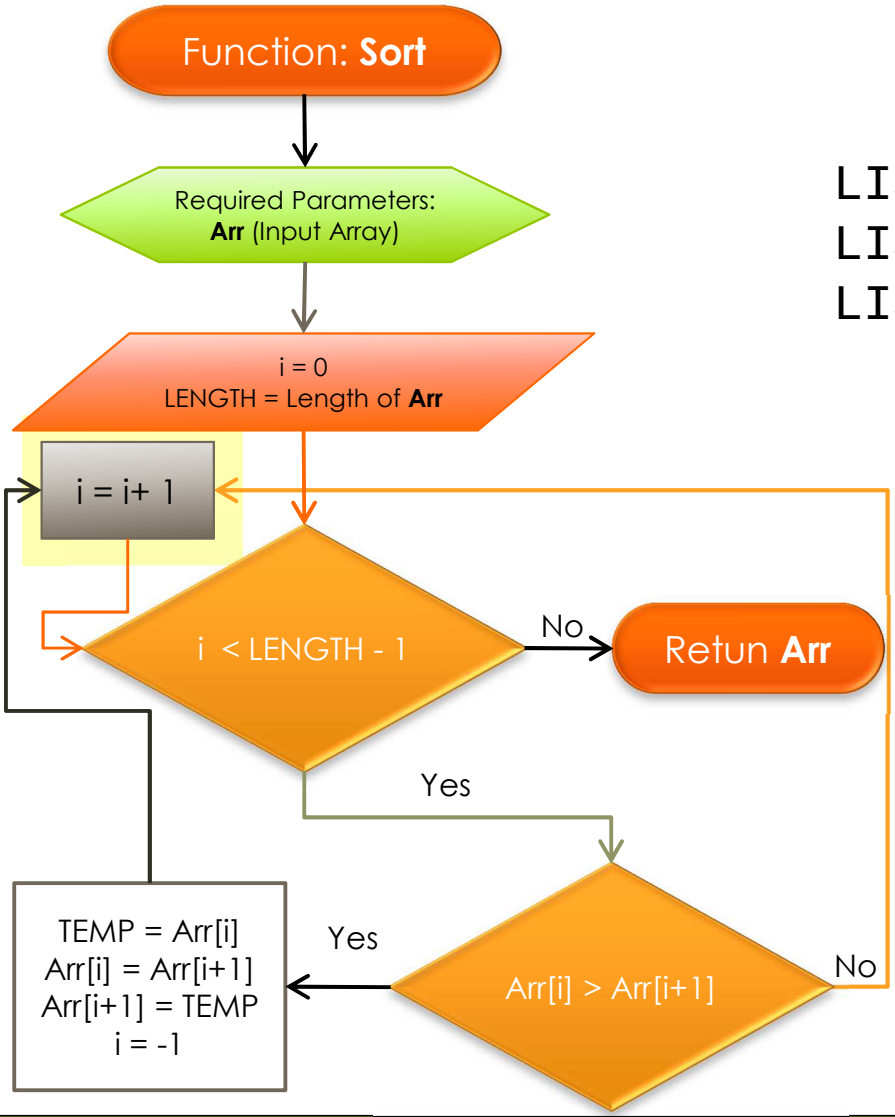


Pseudo Code

```
LIST[0] = 1
LIST[1] = 0
LIST[2] = 2
```

```
i = -1
LENGTH = 3
(2 > 0) = YES
TEMP = 2
```

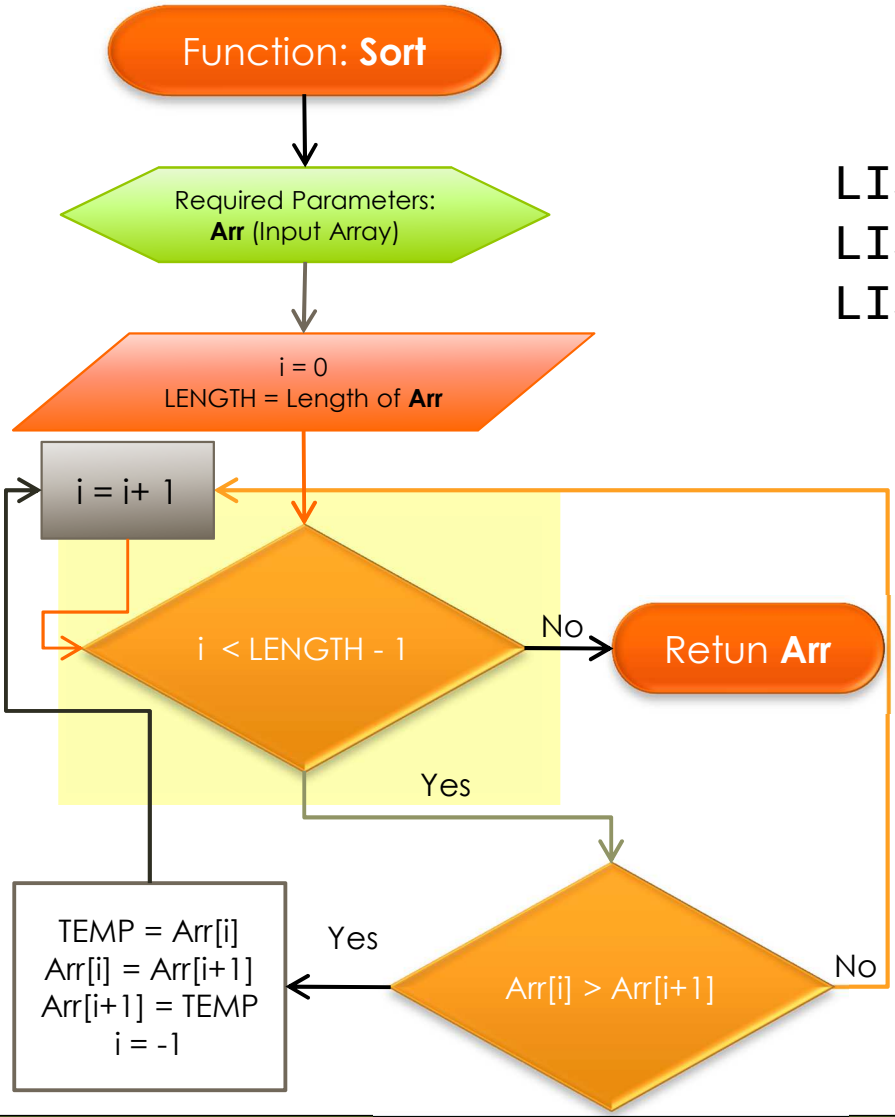
Pseudo Code




```
LIST[0] = 1
LIST[1] = 0
LIST[2] = 2
```

```
i = 0
LENGTH = 3
```

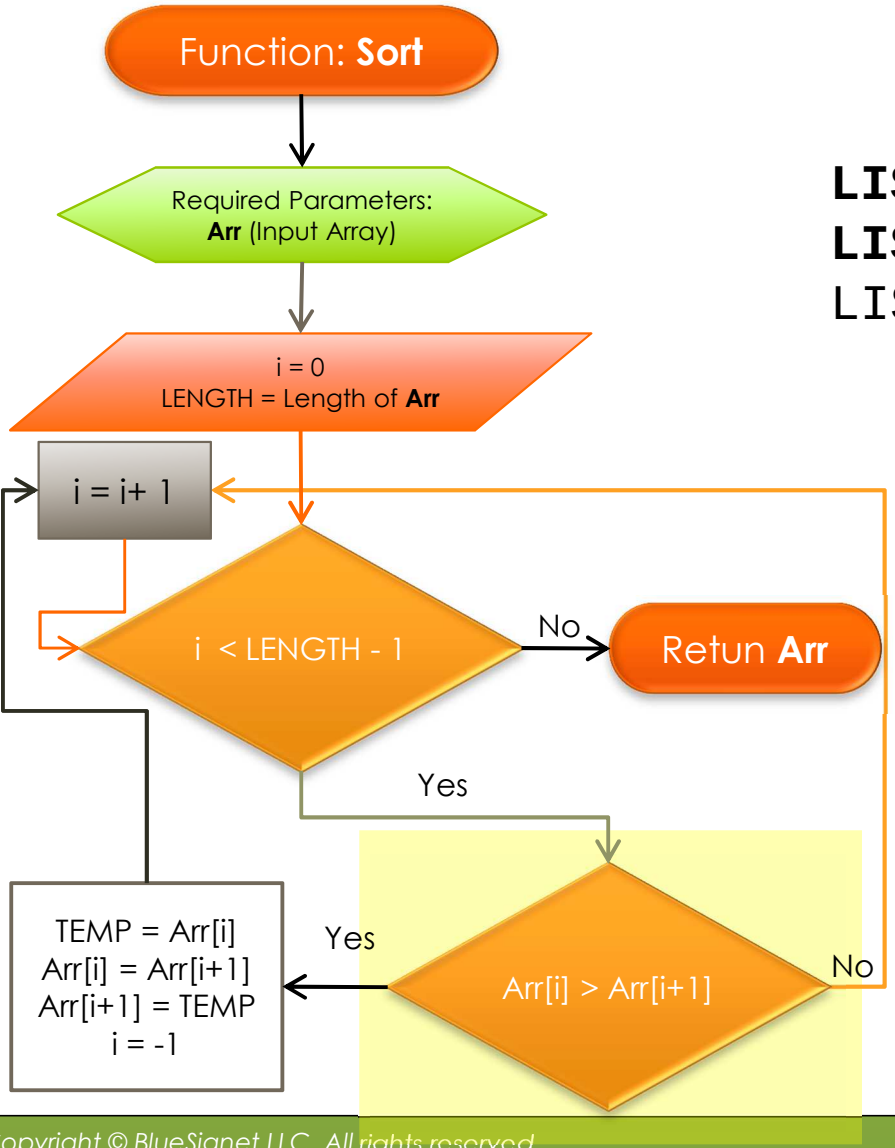
Pseudo Code




```
LIST[0] = 1
LIST[1] = 0
LIST[2] = 2
```

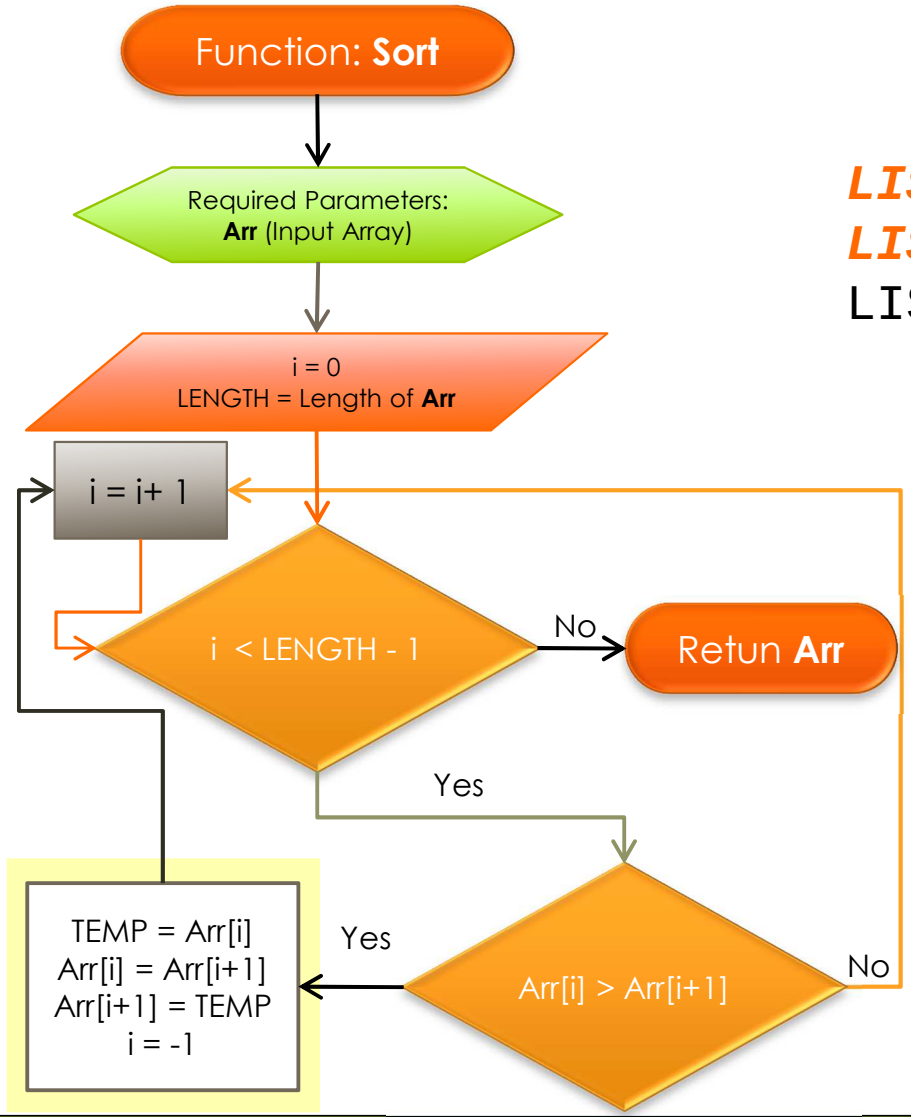
```
i = 0
LENGTH = 3
(1 < 2) = YES
```

Pseudo Code




```
LIST[0] = 1
LIST[1] = 0
LIST[2] = 2
```

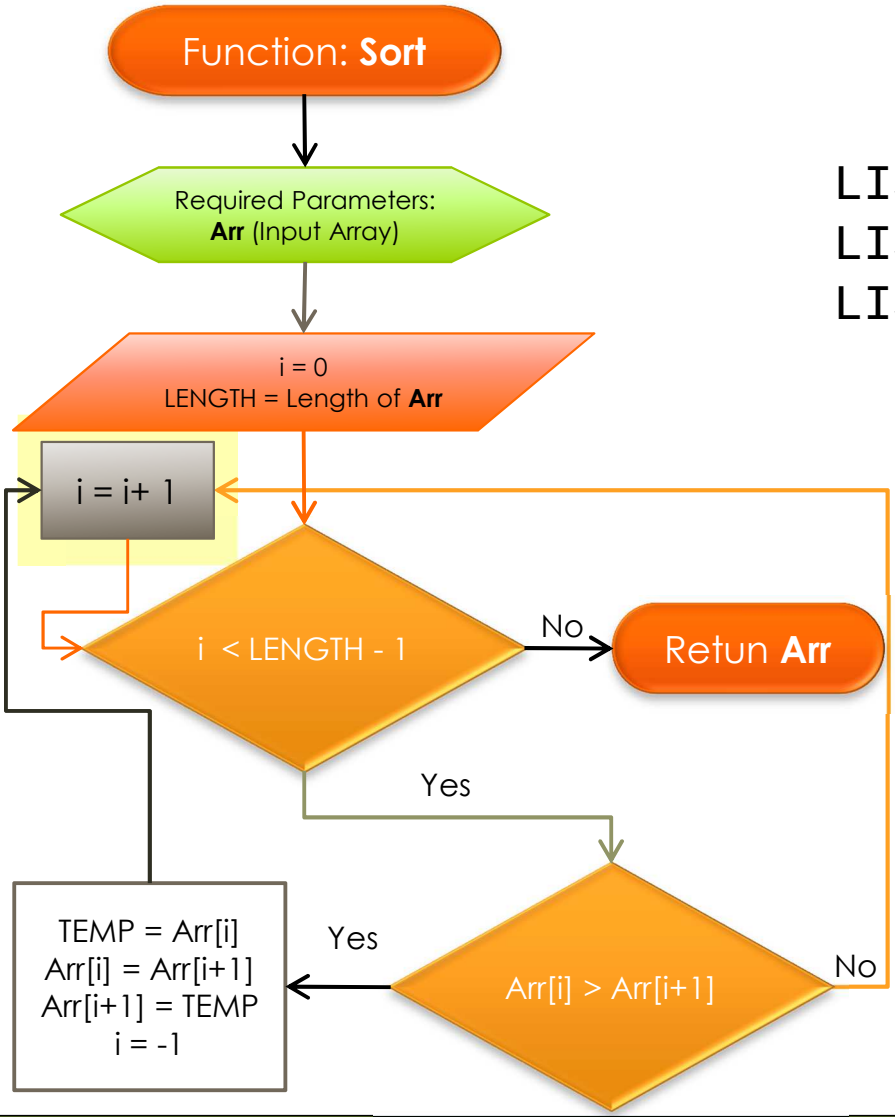
```
i = 0
LENGTH = 3
(1 > 0) = YES
```



Pseudo Code

```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

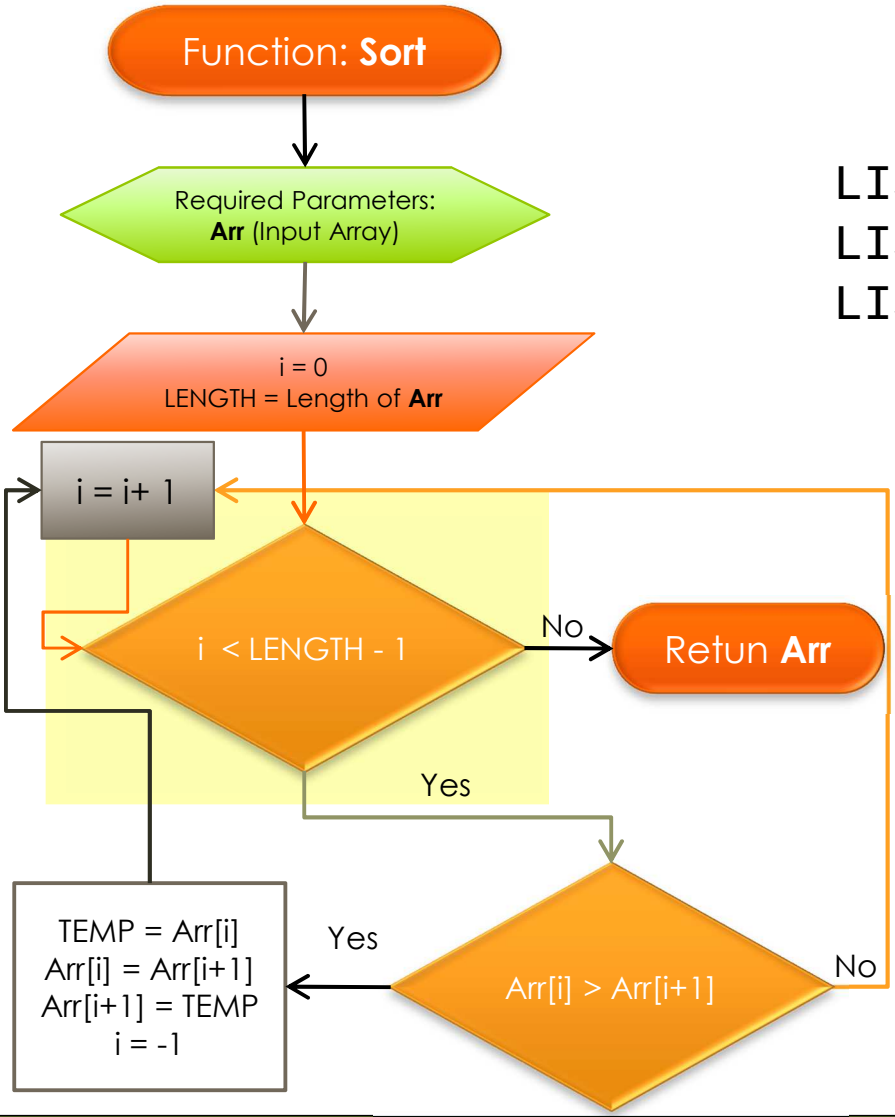
```
i = 0
LENGTH = 3
(1 > 0) = YES
TEMP = 1
```



Pseudo Code

```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 0
LENGTH = 3
```

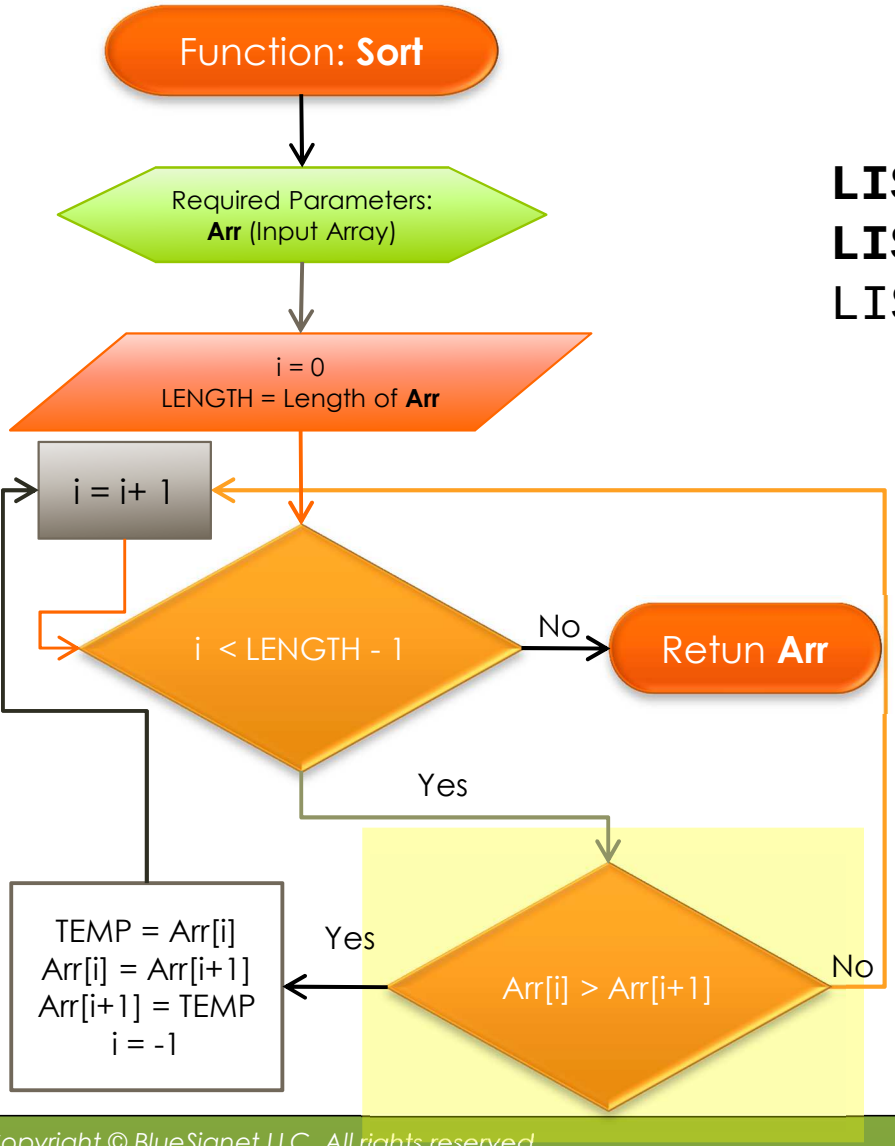


Pseudo Code

```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 0
LENGTH = 3
(0 < 2) = YES
```

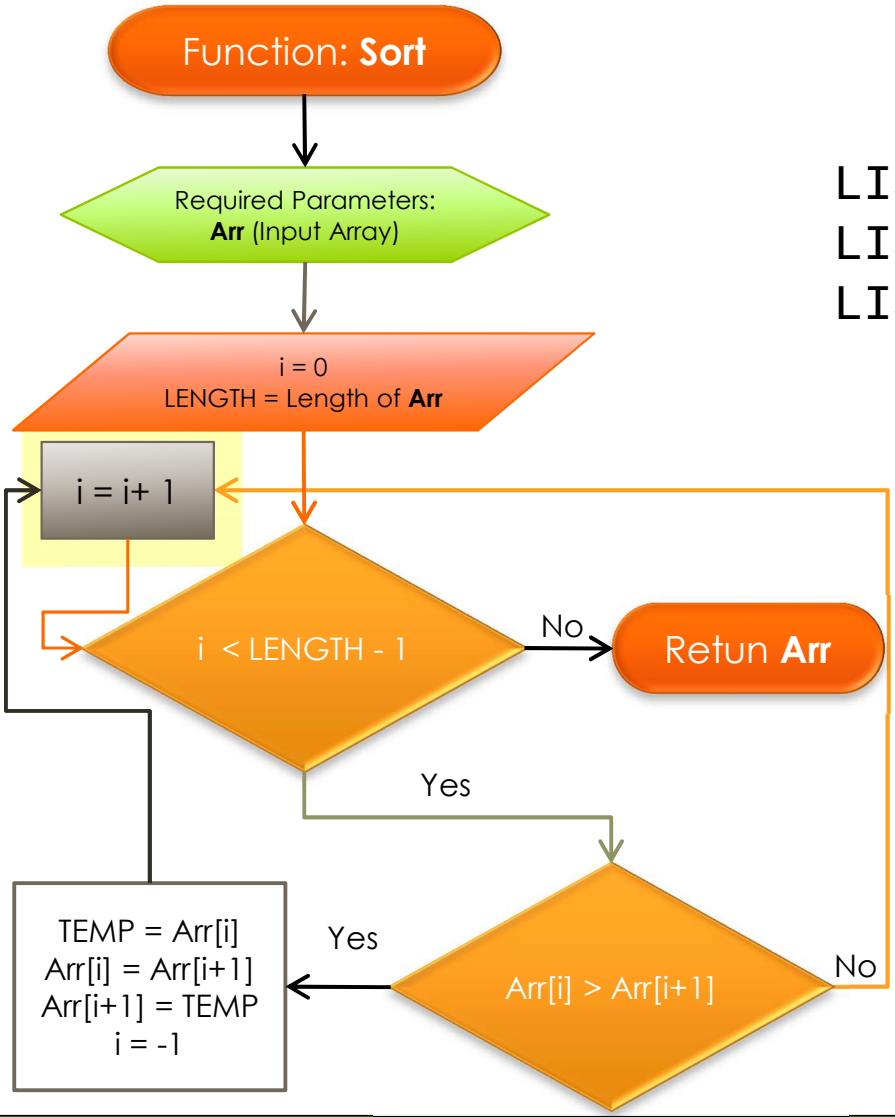
Pseudo Code




```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 0
LENGTH = 3
(0 > 1) = NO
```

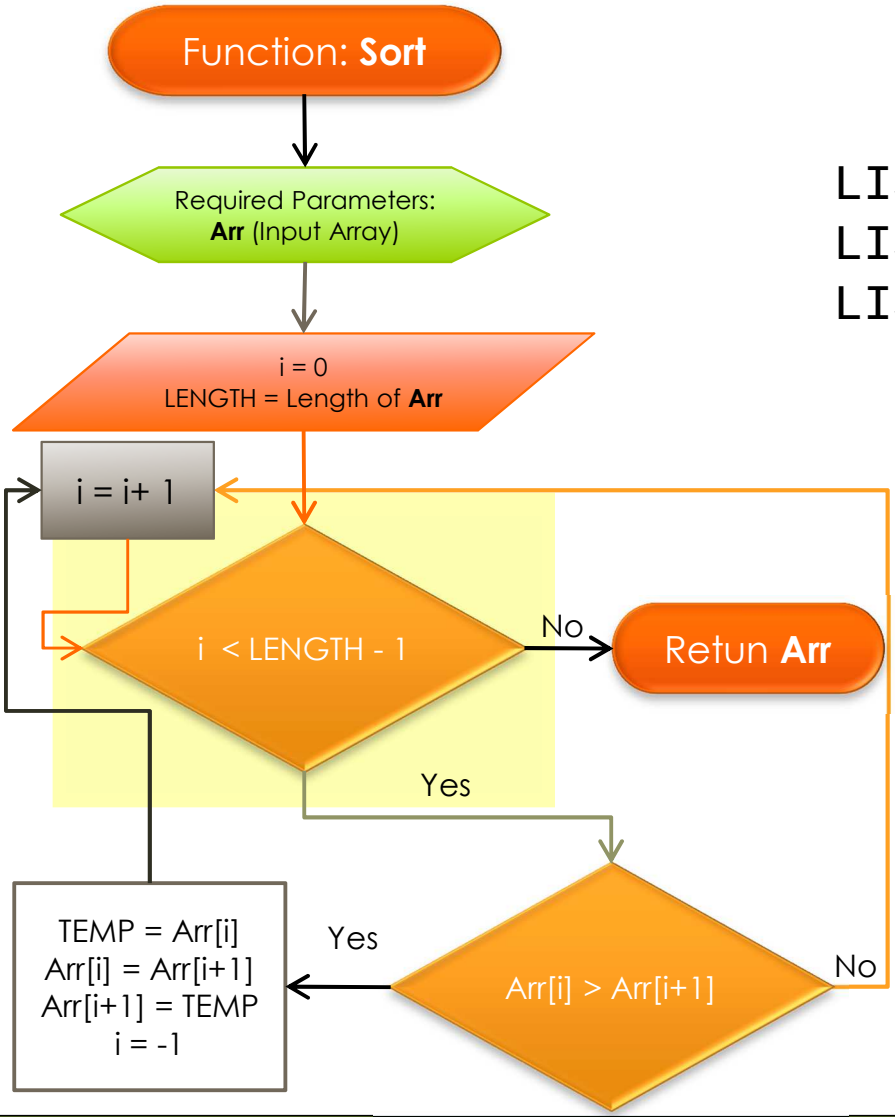
Pseudo Code




```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 1
LENGTH = 3
```

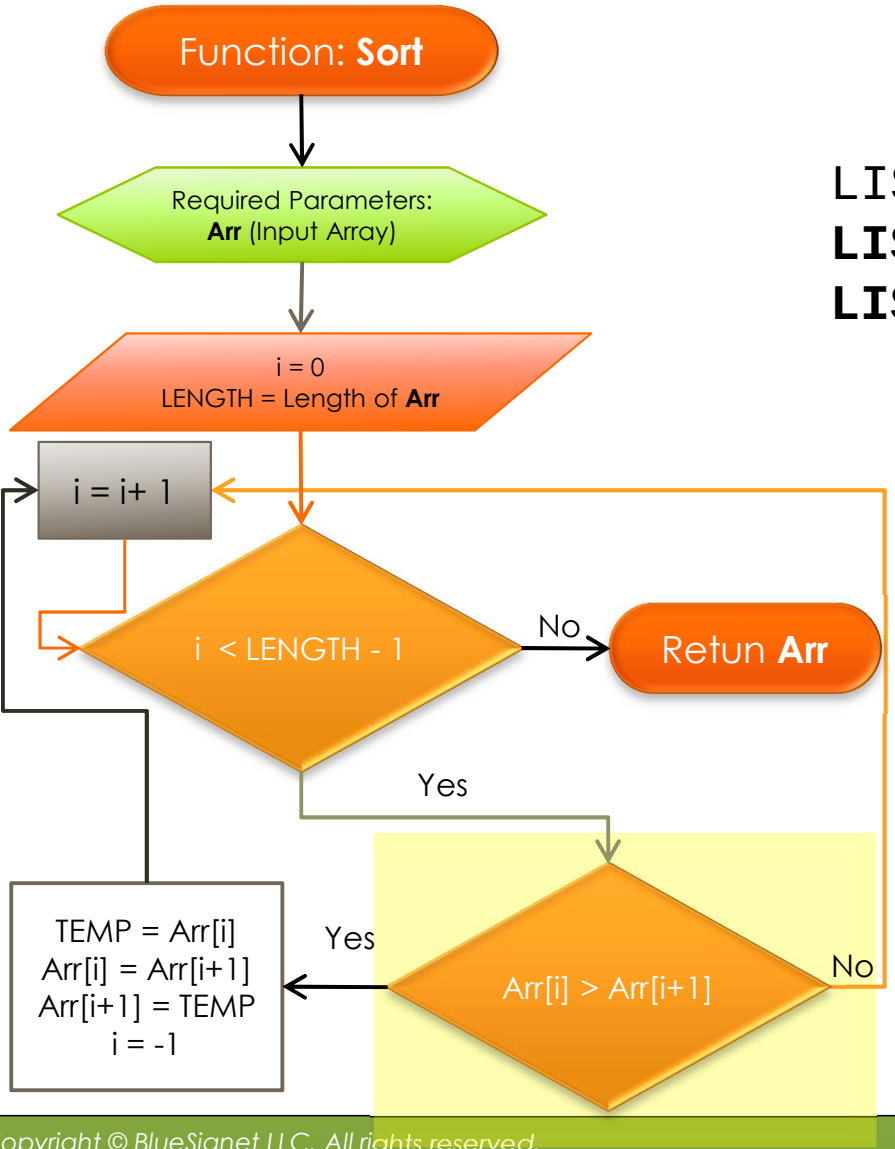
Pseudo Code




```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 1
LENGTH = 3
(1 < 2) = YES
```

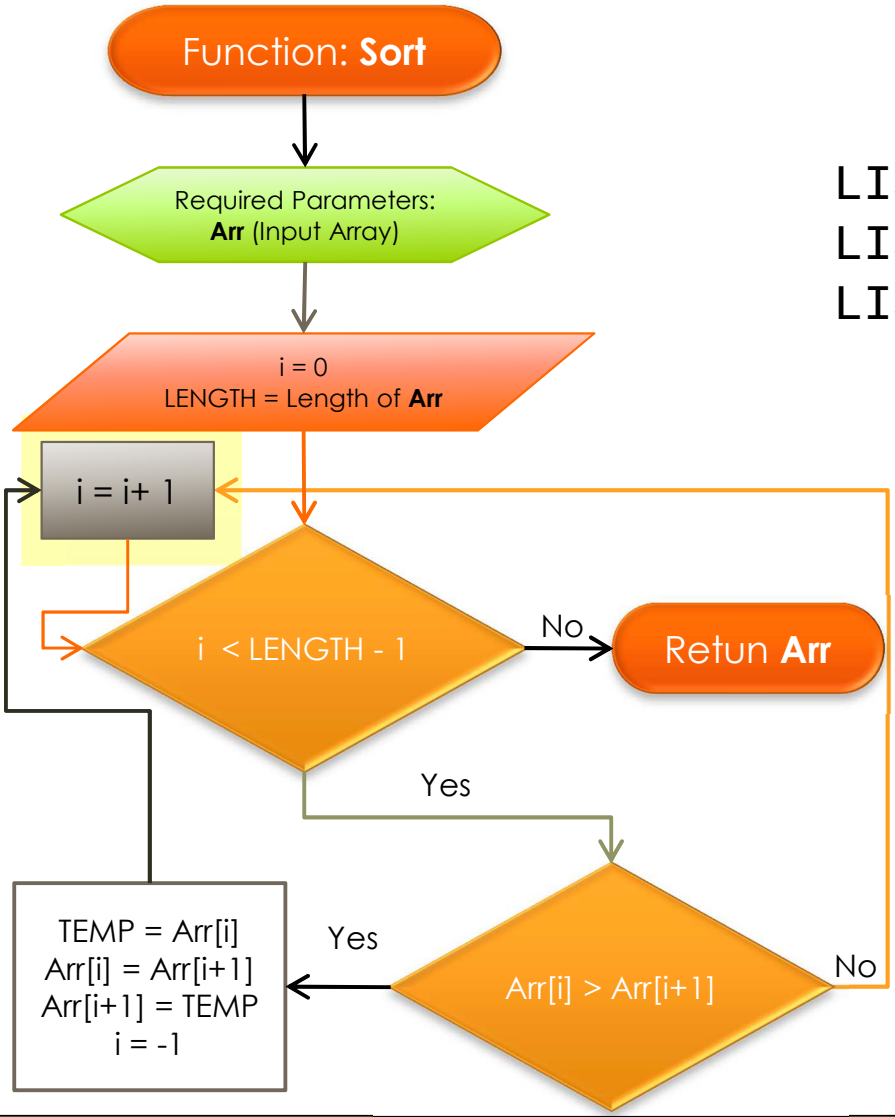
Pseudo Code




```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

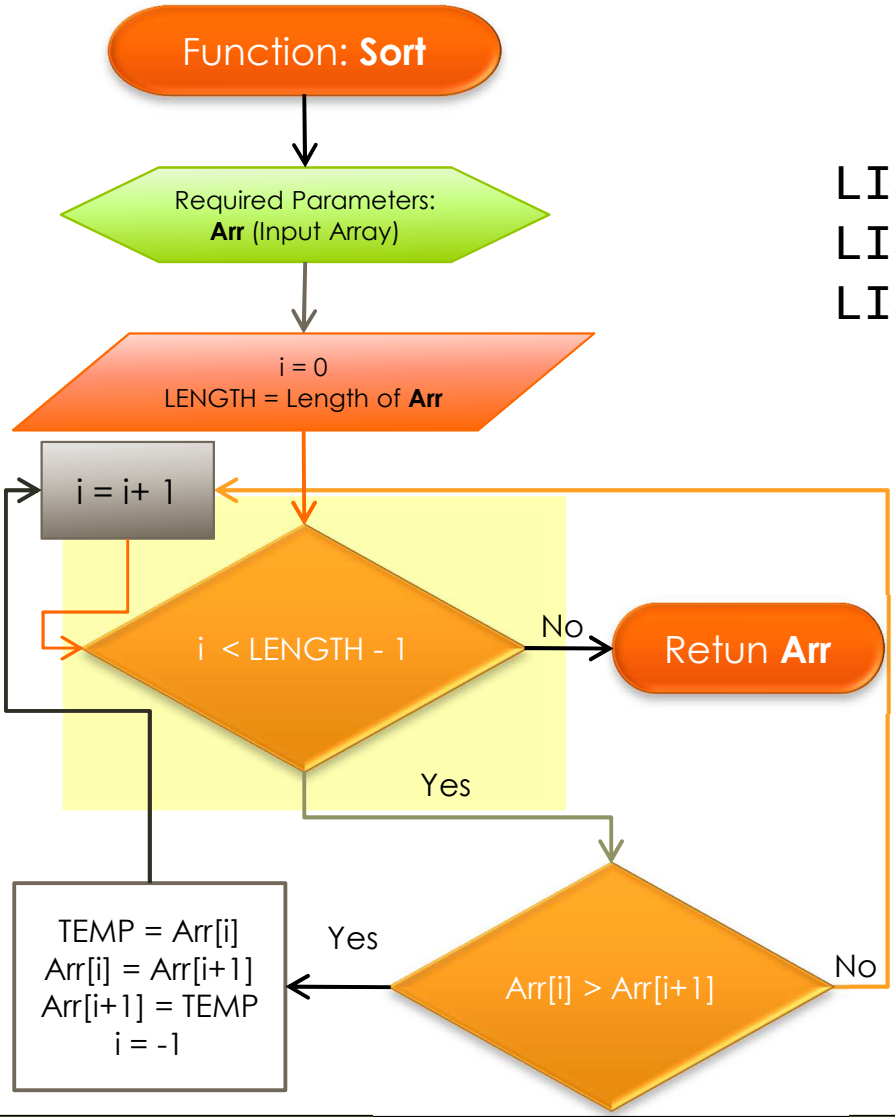
```
i = 1
LENGTH = 3
(1 > 2) = NO
```

Pseudo Code




```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 2
LENGTH = 3
```

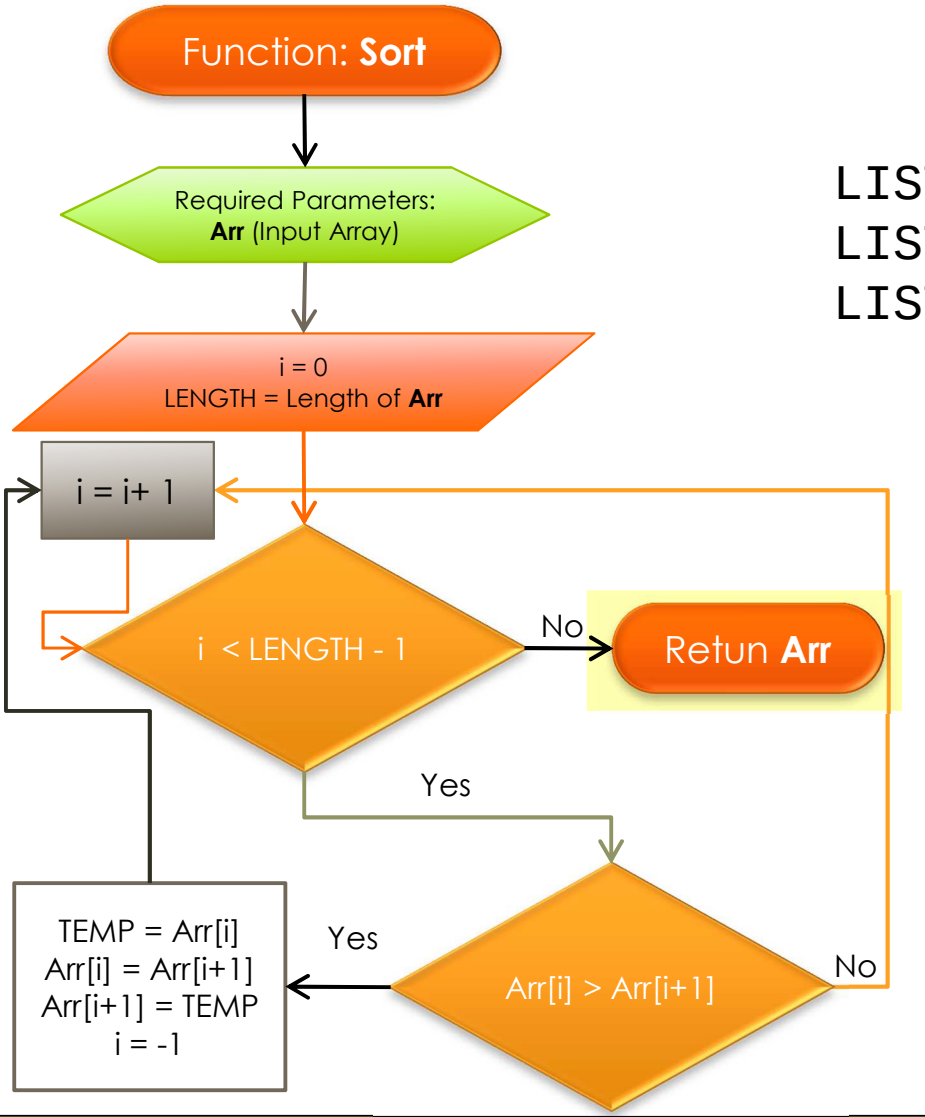


Pseudo Code

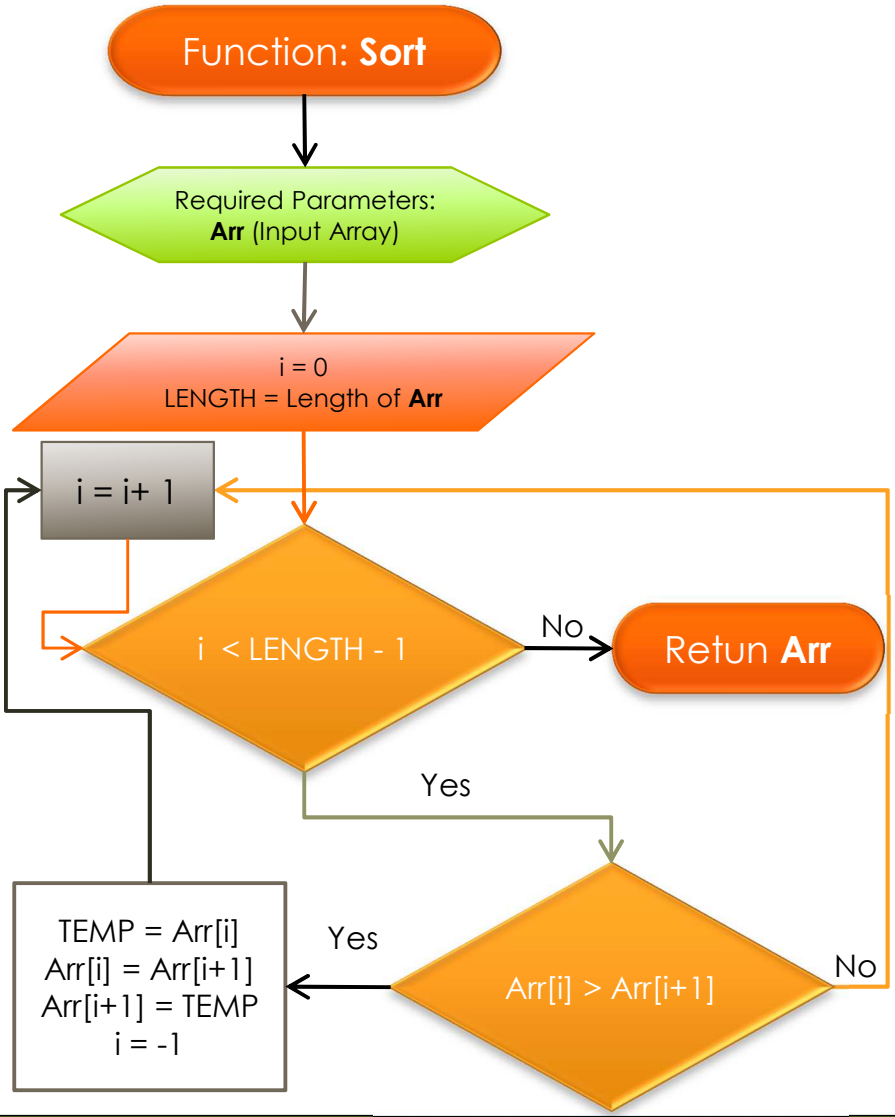
```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```

```
i = 2
LENGTH = 3
(2 < 2) = NO
```

Pseudo Code



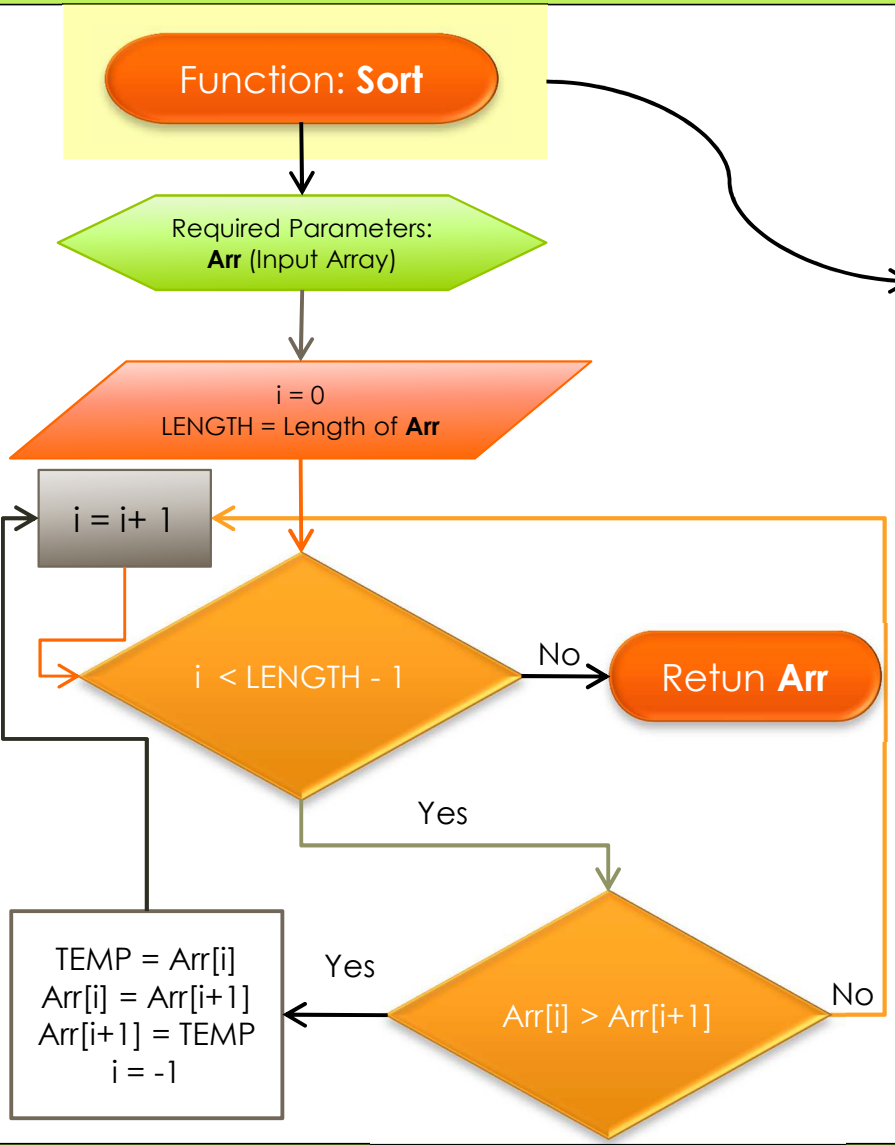

```
LIST[0] = 0
LIST[1] = 1
LIST[2] = 2
```



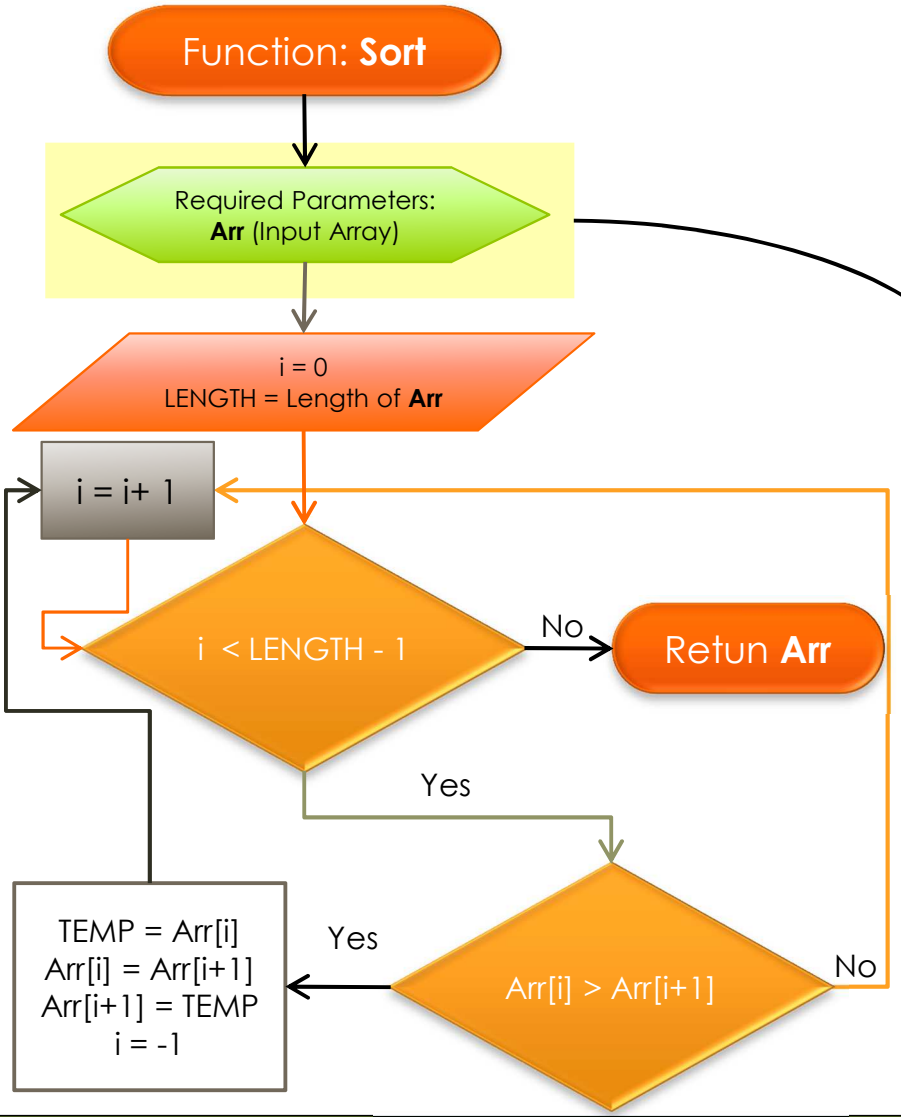
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```

Pseudo Code

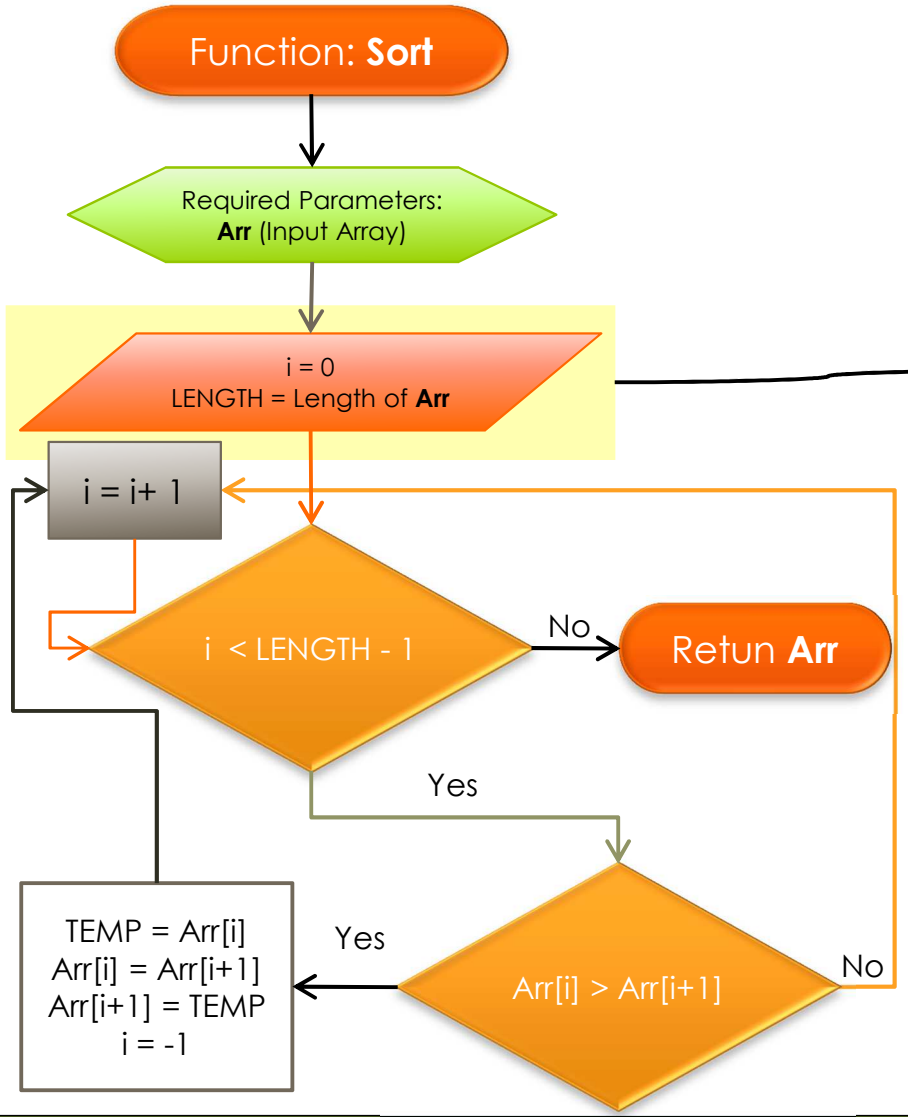



```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



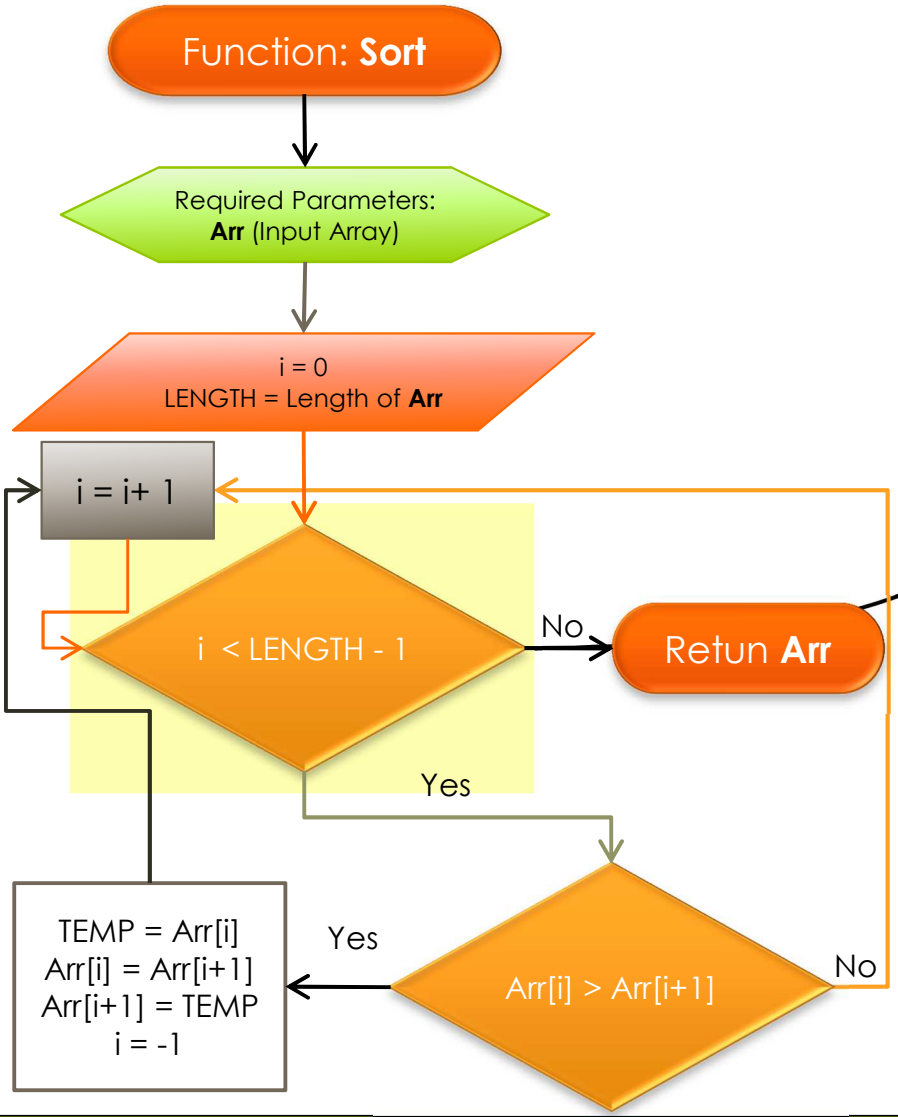
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



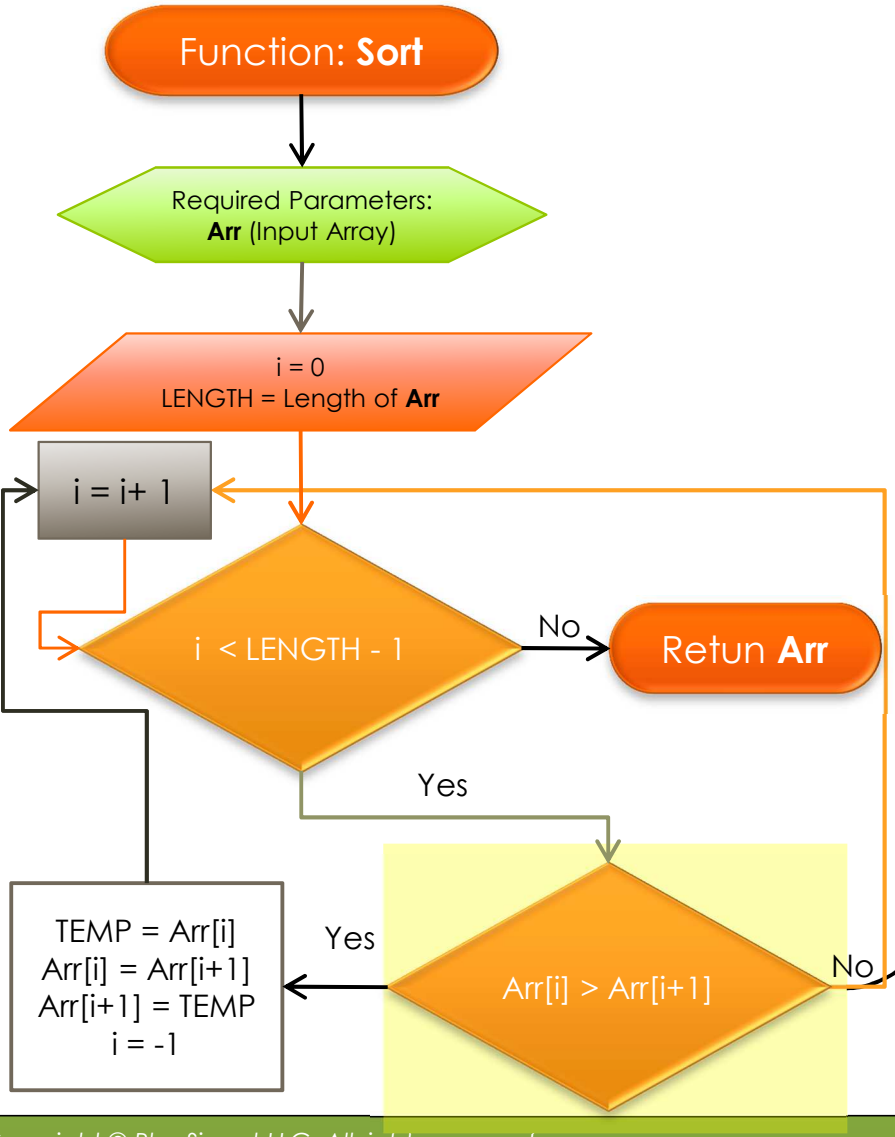
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



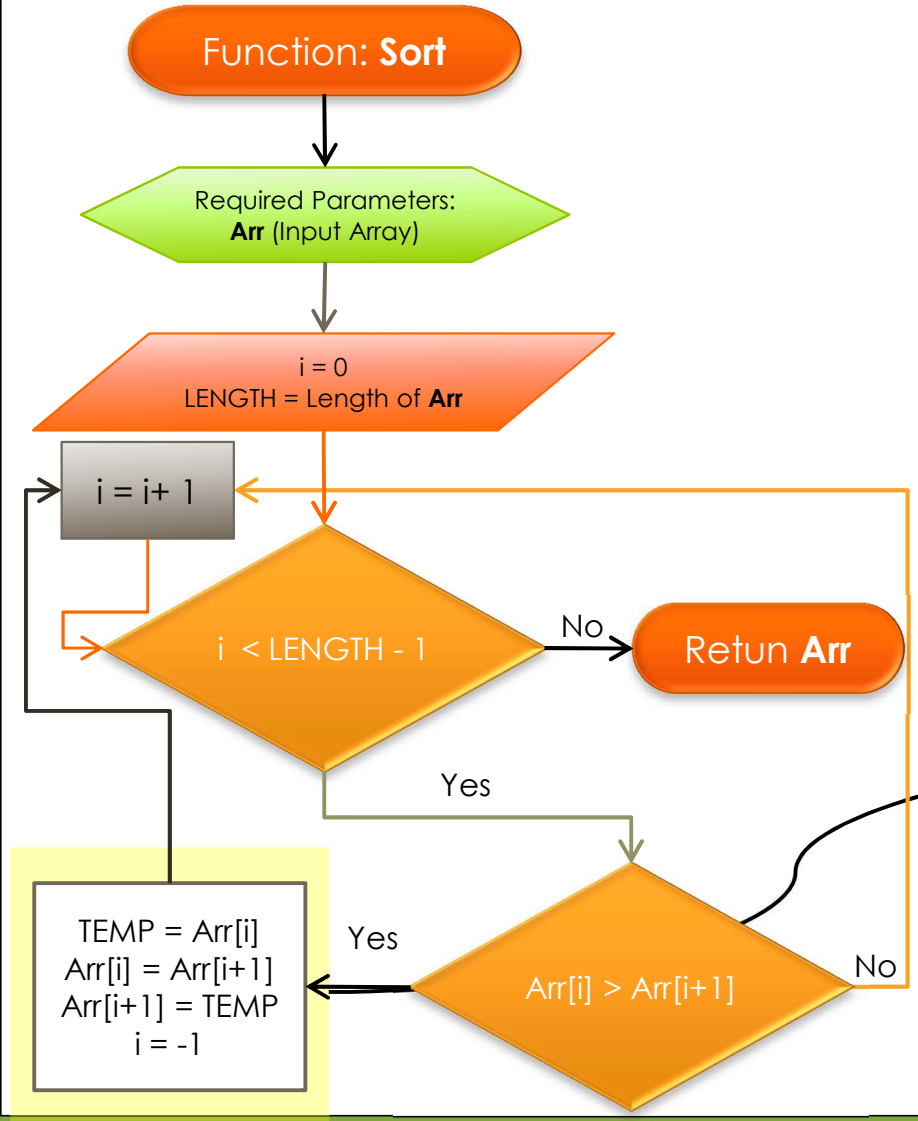
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



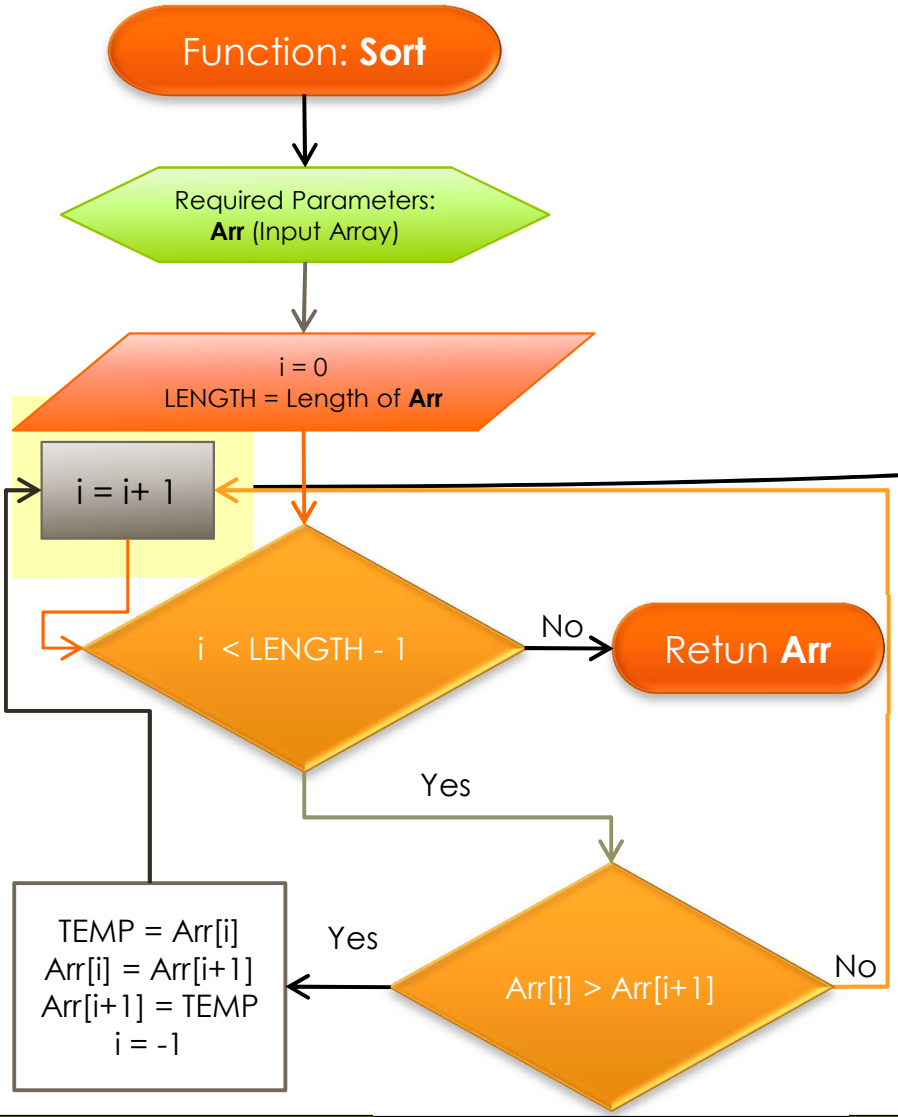
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



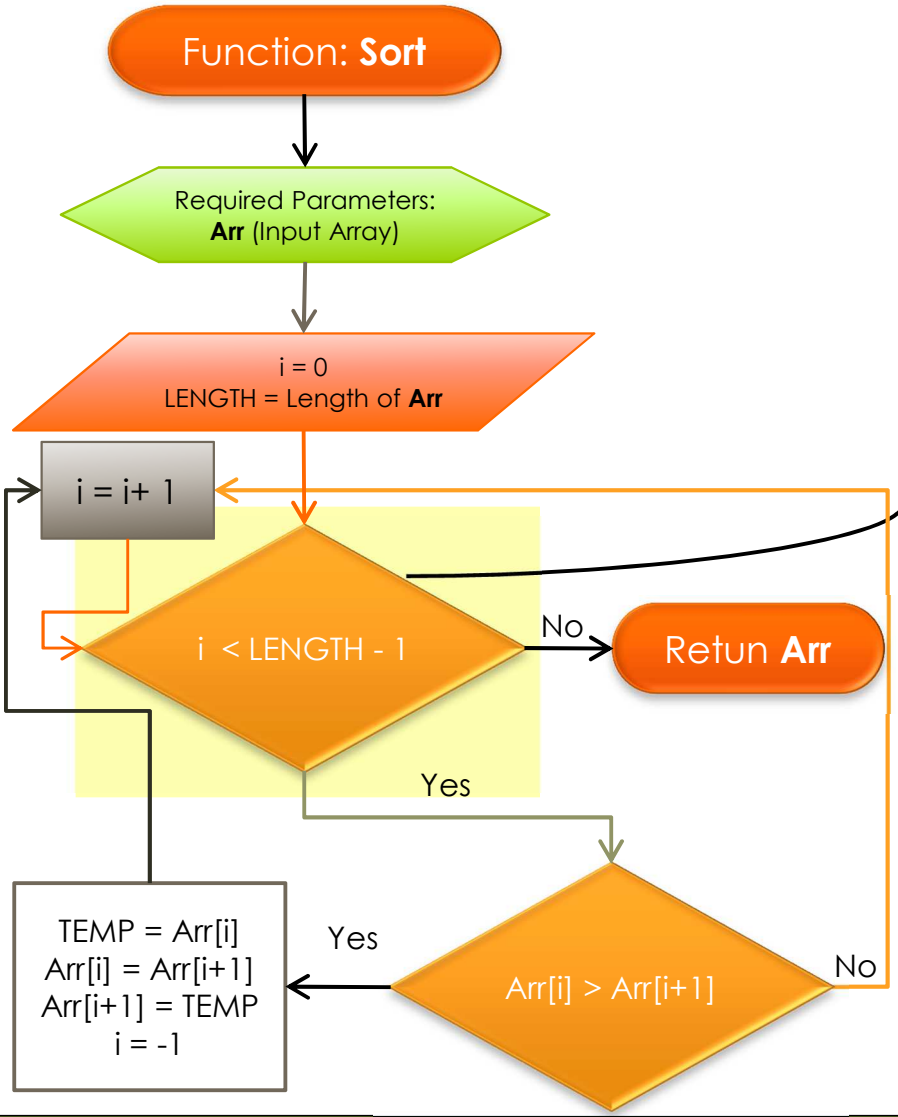
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



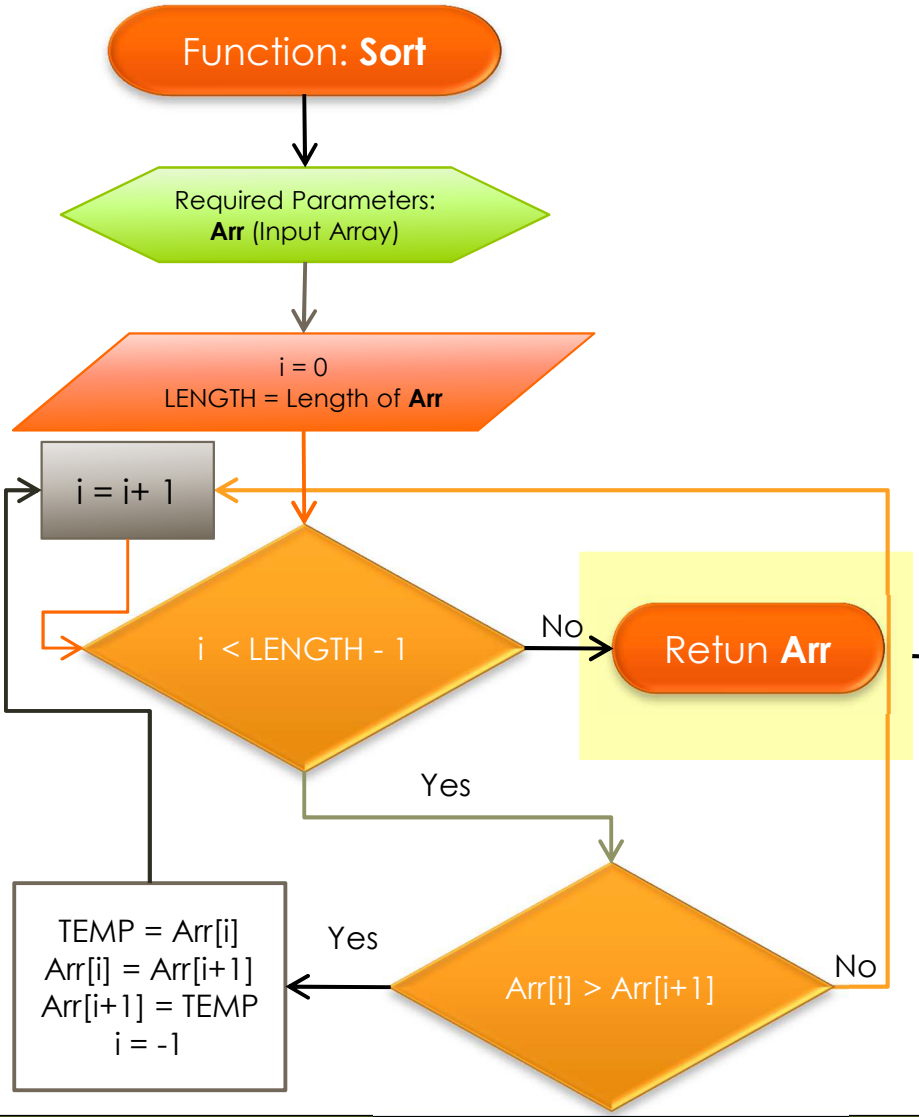
Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```



Pseudo Code

```
Function Sort(Arr)
{
    LENGTH = Length(Arr)
    For (i = 0; i < LENGTH - 1; i++)
    {
        If (Arr[i] > Arr[i + 1])
        {
            TEMP = Arr[i]
            Arr[i] = Arr[i + 1]
            Arr[i + 1] = TEMP
            i = -1
        }
    }
    Return Arr;
}
```

The End?