Assigning, reading, printing or writing a string to a smaller field truncates the string, the tail is lost.

\begin{verbatim}
character(2) :: s = "abcd"  gives s = "ab"
print '(a2)', 'abcd'  prints "ab"
\end{verbatim}

Assigning or reading a string to a longer field adds blank spaces on the right. Short strings are \textit{left-justified} in long fields.

\begin{verbatim}
character(6) :: s = "abcd"  gives s = "abcd  
\end{verbatim}

Printing or writing a string to a longer field adds blank spaces on the left. Short strings are \textit{right-justified} in long fields.

\begin{verbatim}
print '(a6)', 'abcd'  prints "  abcd"
\end{verbatim}
trim(s) trims off any blank spaces at the right end.
trim(" abcd      ") = " abcd"
adjustl(s) left-justifies the string.
adjustl(" abcd") = "abcd  
trim(adjustl(" abcd "))="abcd"
read ' (a) ',t ! reads one line from keyboard to t
read *,r ! reads one word from keyboard to r
read(s,*) r,t ! reads words r, t from string s
write(s,*) r,t ! writes words r, t to string s
If line is 'Good day'
read(line,*) s,t gives s = 'Good' and t = 'day'
read(line,' (a) ') r gives r = 'Good day'
**Character String Operations**

```plaintext
character(5) :: s = 'abc', t = 'xyz'
character(10) :: r

s // t is the concatenation of s and t = s followed by t.

s // t = 'abc  xyz  ',
trim(s) // trim(t) = 'abcxyz'
trim(s) // ' ', // trim(t) = 'abc, xyz'

print'(a,", ",a)', trim(s), trim(t)
write(*,'(a,", ",a)'), trim(s), trim(t)
    prints 'abc, xyz'

write(r,'(a,", ",a)'), trim(s), trim(t)
    gives r = 'abc, xyz'

123456  ! positions in strings start with 1
s = 'abcdef'

s(2:2) = 'b'  ! s(2:2), not s(2)
s(2:3) = 'bc'
s(:3) = 'abc'
s(3:) = 'cdef'

index('abcde', 'cd') = 3
index('abcde', 'cx') = 0
```
CLASSWORK 15.1(2)  name1.f95

!c15_1_2name1.f95
program name1
character(20)::first='John',last='Doe'
print*,'Hello',first,last,'!'  \* Fix the two pinks and add spaces.
endprogram

What went wrong? Correct the two pinks. Add two spaces.
Want  Hello  John  Doe!
not  HelloJohn       Doe   !
not  HelloJohnDoe!

COMMON ERROR  If you don’t see say the last !, it might be off screen due to untrimmed blank spaces.
program name2
character(20) :: fullname
print *, 'Enter first name, space, last name'
read *, fullname
print *, 'Aloha ', trim(fullname), '!
endprogram

Correct the pink line to read in all of the fullname.

Want Aloha John Doe!
not Aloha John
Write a program which reads a first name \texttt{first}, a last name \texttt{last}, then writes full name= last name, \texttt{comma}, space, first name to a strings \texttt{full},full2.

E.g. \texttt{full}= Smith, Tom

```fortran
!c15_3_2name3.f95
program name3
character(20)::first, last, full, full2, full3
full='John Doe'
read(full,*) first, last
first='John'; last='Doe'
full2 = trim(last)  // ________
!Want: Doe, John
print *, full2
write(full3,'(_______)') trim(last),trim(first)
print*,full3  !Above line is an alternate to concatenation.
endprogram
```

Recall

\begin{align*}
\text{index('abcde','cd')} &= 3 \\
\text{index('abcde','cx')} &= 0
\end{align*}
Write a program which, reads a full name full such as full=“Smith, John” consisting of a last name, comma, space, first name, then prints the first name, a space, then the last name (with no commas).

Hint: use the substring functions, e.g., s(:3), full(3:), and the index function i=index(full,','').

Change the 4 pink lines. Start with reads commented out.

!c15_4_2name4.f95
program name4
character(20):: first, last, full
full='Doe, John'
i=index(________) ! the index of ',' in full.
first = __________ ! full(:i) ? full(i:) ?
last = __________ ! full(:i) ? full(i:) ?
print *,first,last !fix three things:
!Want: John Doe no comma
endprogram
**COMMON error**  Don’t set first=index(full,'',''). first is a character string, i=index(...) is an integer. Set first = to some substring of full, maybe full(i:)? -- actually not.

**ASCII numbers**

Letters have numbers, ASCII numbers.
Letters: “A”, “B”, “C”, ..., “a”, “b”, “c”, ...
ASCII : 32, 33, 34, ..., 64, 65, 66, ... ,
If let is a letter, iachar(let) is its ASCII number.
If n is a number, achar(n) is its ASCII character.
Hence iachar(“A”) = 32 and achar(32) = “A”
Write a subroutine `lower_case(u)` which converts a phrase `Hello There` to all lowercase letters `hello there`. Fill the pink blank.

```fortran
subroutine lower_case(word)
  character(*) :: word
  do i = 1, len(word)
    select case(word(i:i))
      case("A":"Z")
        word(i:i)=________________
    endselect
  enddo
end subroutine

program test_lower_case
  character(30) :: word="AaBbZz"
  print*, trim(word)
  call lower_case(word)
  print*, trim(word)
end program
```
The next homework problem needs a subroutine inverses_letters(word,n,backword) word word of n
letters, makes a word backword with the letters in the reverse order. If word is "abcde" then backword is "edcba". The two example subroutines below work only for a word of length n=3, your subroutine must work with words of any length n. The two alternatives have equal points.

Recall word(i:i) is the i\text{th} letter.

!reverse_letters2a.f95
subroutine reverse_letters(word,n,backword)
character(n)::word,backword
backword=""
backword=trim(backword)//word(3:3)
backword=trim(backword)//word(2:2)
backword=trim(backword)//word(1:1)
endsubroutine

!reverse_letters2b.f95
subroutine reverse_letters(word,n,backword)
character(n)::word,backword
write(backword,'(200a1)')word(3:3),word(2:2),word(1:1)
endsubroutine
Do one of **Homework 15.1a** or **Homework 15.1b** not both.

**Homework 15.1(5)** reverse_lettersa.f95

e-mail: dale@math.hawaii.edu  s-ubject line: 190 h15.1a(5)

To generate a sum, you start with 0 and add items one at a time, \( S=S+\ldots \). To generate a word, start with "" and concatenate one letter at a time \( \text{word}=\text{trim(\text{word})} //\ldots \).

!reverse_lettersa.f95
!subject line: 190 h15.1a(5)

subroutine reverse_letters(word,n,backword)
character(n)::word,backword

!delete this, replace with an initialization assignment
!followed by a three line do-loop
endsubroutine

program test_reverse_letters
character(6)::backword,word='123abc'
call reverse_letters(word,6,backword)
print*,word," ",backword
endprogram

Do one of **Homework 15.1a** or **Homework 15.1b** not both.
Homework 15.1(5) reverse_lettersb.f95

email: dale@math.hawaii.edu subject line: 190 h15.1b(5)

Replace the deleted line with one write statement which writes the word word into the word backword. You’ll need a format of '(200(a1))'. Use an implicit do statement such as print*, (word(i:i), i=3, 1, -1)

!reverse_lettersb.f95
!subject line: 190 h15.1b(5)
subroutine reverse_letters (word, n, backword)
character (n):: word, backword

!delete this, replace with one write(line, ... statement
endsubroutine

program test_reverse_letters
character (6):: backword, word='123abc'
call reverse_letters(word, 6, backword)
print*, trim(word), " ", trim(backword)
endprogram

□
The next homework problem needs a subroutine `line_writer(words,numwords,line)` which writes a vector of words, e.g., `words=["one","two","three"]` of length `numwords` into a single line `line="one two three"` with the words separated with spaces. The subroutines below work only for vectors of length 2, your subroutine must work with vectors of any length. The two alternatives have equal points. All strings have at most 200 characters.

!line_writer2a.f95
subroutine line_writer(words,numwords,line)
character(200)::words(numwords),line
line=""
line=trim(line)//" "//trim(words(1))
line=trim(line)//" "//trim(words(2))
endsubroutine

!line_writer2b.f95
subroutine line_writer(words,numwords,line)
character(200)::words(numwords),line
write(line,'(200(a," "))') trim(words(1)),trim(words(2))
endsubroutine
Do one of **Homework 15.2a** or **Homework 15.2b** not both.

**Homework 15.2a(6) line_writera.f95**  
daile@math.hawaii.edu  subject line: 190 h15.2a(4)

To generate a sum, you start with 0 and add items one at a time, $S=S+...$. To generate a word, you start with "" and concatenate items one at a time $\text{word}=\text{trim(\text{word})} //...$.

**line_writera.f95**  
**subject line: 190 h15.2a(4)**

```fortran
subroutine line_writer(words,numwords,line)
character(200)::words(numwords),line
integer::numwords
    !delete this, replace with an initialization assignment
    !followed by a three line do-loop
endsubroutine

program test_line_writer
character(200)::words(200),line
words(1:5)=(/"aa","bb","cc","dd","ee"/)
do i=2,5
call line_writer(words,i,line)
print*,"The line is: ",line
endo
endprogram
```
Do one of **Homework 15.2a** or **Homework 15.2b** not both.

**Homework 15.2b(6) line_writerb.f95**
dale@math.hawaii.edu  **subject line: 190 h15.2b(4)**

Replace the deleted line with one write statement which writes the vector `words` into the line `line`. You'll need a format of `' (200(a," "))'`. Use a implicit do statement like

```fortran
print*, (trim(words(i))," ",i=1,numwords)
```

!line_writerb.f95
!subject line:  190 h15.2b(4)
subroutine line_writer(words,numwords,line)
character(200)::words(numwords),line
integer::numwords
  !delete this,  replace with one write(line, ... statement
end subroutine

program test_line_writer
character(200)::words(200),line
words(1:5)=(/"aa","bb","cc","dd","ee"/)
do i=2,5
call line_writer(words,i,line)
print*,"The line is: ",line
endo do
end program
Homework 15.3(2) upper_case.f95 Write a subroutine upper_case(u) which converts a phrase Hello There to all uppercase letters HELLO THERE.

email: dale@math.hawaii.edu subject: 190 h15.3(2)

!upper_case.f95
!subject line: 190 h15.3(2)

subroutine upper_case(word)
    !delete this, finish the subroutine
endsubroutine

program test_upper_case
character(30)::word="AaBbZz"
print*,trim(word)
call upper_case(word)
print*,trim(word)
endprogram