report

Jiachen Liu

1.

a.

The first obstacle I met was to get the program to read the file. For some reason the while sentence does not work on the Xcode for me so I decided to use infile.get(c), where c is a character. I have to use infile.clear(); and infile.seekg to set a new position for the stream (in my case, the starting point).

b.

The other obstacles is to try to get it print out exactly as the project spec stated. There are a lot of conditions to consider, especially, adding the spaces and taking the ‘-’ into account is really hard for me. But I overcame it with my cleverness.

2.

The function mainly consists of these functions:

The initializer and terminator will not be used in the actual run. They are for starting the program, by opening files and checking files for openness and so on.

It looks like:

void initializer ()

{

Open the two files.

If they are not opened successfully, output an error message.

}

void terminator ()

{

close the two files.

If they are not closed successfully, output an error message.

}

void processArr1(some parameters)

{

use a for loop. clear up the extra spaces in the text.

Use a for loop, clear up the extra lines in the text.

}

void processArr2 (some parameters)

{

use a for loop add a space after every ‘.’, ‘!’ or ‘?’.

}

void print (some parameters)

{

**a big while loop, below are all under the while loop**

{

because I used processArr1 and processArr2, there are some unwanted result due to these function, I use if condition to clear them up first.

Use if statement to take account when #P# is in the front or the end of the page.

Use a while loop to count the wordlength left (the left length of the word we are printing)

If wordlength is larger than the line length, return 1. Since it’s void, I’ll store the number into a reference parameter.

If (currentlength+storedlength<=maxlength)

{

Outf each char and add it’s currentlength by 1.

}

Else

{

It can be an extra space, so we take that into account

Or it can be the wordlength larger than maxlength.

Or it can be a new line. (outf new line)

}

}

}

int pack(int lineLength, istream& inf, ostream& outf)

{

if the linelength smaller than 1, return 2.

Use a while loop to counter the number of characters here

Reset the istream location.

Use a while loop to put the characters into an array.

Use processArr1

Put the edited array into a new array.

processArr2

then use the print function and return corresponding conditions.

}

3. test conditions:

Here’s the list of test data:

Put #P# in the front:

test whether the function still output it as a extra line

put #P# in the end:

test whether the function still output it as a extra line

put #P# in the front and #P# in the middle

it does work, but the project 5 packer online doesn’t work so well. So not sure if I’m wrong.

Put multiple #P# in the middle

To see if it only output one.

Put multiple #P# in the front

To see if it outputs none.

Put multiple #P# in the back

To see if it outputs none. **No it the program does not work if it put multiple #P#s in the back.**

Put -1 as the linelengtth:

See if the function’s value is 2.

Put more than 400 for linelength:

To see if it works, it works.

Put a hypen between words as the end of the line:

To check if the word portion works. It works.

Put multiple hypens between words as the end of the line:

To check if the word portion works. It works.

Change the linelength to 40 see if it perfectly fits the online packer. The test being a sample portion given by the professor.

A regular test. Yes it fits. And to see if it “fits as much as possible”.

Change the linelength to 10 see if it perfectly fits the online packer.

See what happens if the linelength is small. Yes it fits.

Change the linelength to 100 see if it perfectly fits the online packer.

See what happens if the linelength is really big. Yes it fits.

Put a really lone word in the middle of the sentences

To check if the word portion works. It works.

Put a really lone word in the middle of the sentences, the word is longer than the line length.

To check if it breaks the word down. Yes it breaks it down and output 1.

Only put one word as the whole essays (>300 chars) to see if it works

A very extreme case.

To put a word one space after at the end of the line:

To see if it ignores the space when counter the linelength.

To put a word with spaces after at the end of the line:

To see if it ignores the space when counter the linelength.

Put one empty line:

see if the output line is followed by any empty lines. No it doesn’t.

Put multiple empty lines:

see if the output line is followed by any empty lines. It does not work so well.

……etc.