Cynthia Nguyen 705767700

CS 31, Project 3, hero.cpp

Pseudo code:

bool hasProperSyntax(string tune)

{

empty strings has proper syntax;

first character cant be digit;

last character has to be ‘/’;

every ‘/’ is considered proper syntax;

turn every letter lowercase, only letters “g,r,y,b,o” are considered proper syntax;

notes have to be followed by a digit or a slash;

digits cant follow after a ‘/’;

if two digits are next to each other, next char has to be a ‘/’;

every other char is considered improper syntax;

}

int convertTune(string tune, string& instructions, int& badBeat)

{

Check for proper syntax;

Loop for each char in string;

Check if char is a note;

Note is not sustained;

{

Increment beat;

Append lowercase note to string;

Skip one char;

}

Note is a sustained note;

If length of sustained note is less than 2, return 3;

{

Make a substring for the length of the sustained note

Loop through substr

return 2 & set badBeat to incorrect beat if there is a

note where there is supposed to be a beat;

return 4 if beat ends prematurely;

If there is correct number of beats;

append uppercase note to string;

increase beat;

Skip all chars of sustained note;

}

If char is a beat (‘/’) not part of a note or sustained note;

Append ‘x’ to string;

If loop completes, set instructions, return 0;

}

Obstacles:

* hasProperSyntax
  + I couldn’t check the character before a digit if the first character is a digit
    - If the first character is a digit, it is already not proper syntax, so I set a statement to rule out those scenarios before needing to call for a character before a digit
  + I couldn’t check if the character after a note or digit is a ‘/’ if it is the last character
    - To be have proper syntax, the last character would have to be a ‘/’ for all cases, so I set a statement to rule out all strings without a ‘/’ at the end before calling to check for a character after a digit or note
  + I did not understand that !=(Boolean) && !=(Boolean) is the opposite of ==(Boolean) || ==(Boolean), and had some statements run when they shouldn’t have
* convertTune
  + Put a statement as a “if” statement rather than an “else if” statement, and created instructions that did not match the proper conversion of the tune
  + Convert tune kept returning an incorrect value
    - Was an issue in indexing parameters, caused the conversion from character digits to integers to be converting the note to an integer
  + Instructions kept having double the amount of notes for every sustained note
    - Calling my SustainedNote function twice caused the issue because the instructions are passed by reference

Test Cases

//checks that returns true for proper syntax

assert(hasProperSyntax("g/b//"));

//checks for non note letters

assert(!hasProperSyntax("g/z//"));

//checks that every note is followed by a beat

assert(hasProperSyntax("r/"));

assert(!hasProperSyntax("r"));//cant check whats after the last char, solved by requiring last char == '/'

//checks empty strings

assert(hasProperSyntax(""));

//checks digits needs to follow notes

assert(hasProperSyntax("O3///b/"));

assert(!hasProperSyntax("g/b/3/"));

assert(hasProperSyntax("g/b3///"));

assert(!hasProperSyntax("03///b/"));//cant check whats before first char, solved by requiring first char cant be digit

//checks that max is two digits

assert(!hasProperSyntax("g/b/b300/"));

//checks that tune can consist of only '/'

assert(hasProperSyntax("/"));

//checks that both uppercase and lowercase are considered proper syntax

assert(hasProperSyntax("G/b//O/y2//"));

string instrs;

int badb;

badb = -999; // so we can detect whether this gets changed

instrs = "WOW"; // so we can detect whether this gets changed

//checks if ChartoInt function successfully converts digit characters of a string into integers

string s = "g3///";

assert(CharToInt('0', s.at(1)) == 3);

assert(CharToInt('1', '0') == 10);

//tests function SustainedNote to print out uppercase notes for number of beats its sustained

string test;

int beat = 0;

assert(SustainedNote("///", 'g', 3, test, beat, badb) == 0 && test == "GGG" && beat == 3 && badb == -999);

//tests that function returns 1 when propersyntax is false and instrs and badb remains the same

instrs = "WOW";

assert(convertTune("r", instrs, badb) == 1 && instrs == "WOW" && badb == -999);

//checks badbeat outputs properly in return 2

assert(convertTune("r/y3//g/r/", instrs, badb) == 2 && instrs == "WOW" && badb == 4);

badb = -999;

//test for return 3 if sustained note is less than 2 and badbeat outputs properly

assert(convertTune("r/o/g0/b/", instrs, badb) == 3 && instrs == "WOW" && badb == 3);

//tests for return 4 if beat ends prematurely and badbeat outputting properly

badb = -999;

instrs = "WOW";

assert(convertTune("g3//", instrs, badb) == 4 && instrs == "WOW" && badb == 3);//compile error, badbeat and the end of the tune leads to overflow; add if statement saying if last character and not reached number of beats in hold, return 4 and badbeat

//checks for proper conversions to instructions

badb = -999;

assert(convertTune("r/o/g/y/", instrs, badb) == 0 && instrs == "rogy" && badb == -999);

assert(convertTune("r//g/", instrs, badb) == 0 && instrs == "rxg" && badb == -999);

assert(convertTune("g3///", instrs, badb) == 0 && instrs == "GGG" && badb == -999);//instrs == "GGGGGG", solved: calling SustainedNote function twice appended instructions twice because passed by reference

assert(convertTune("g/G/g/B/", instrs, badb) == 0 && instrs == "gggb" && badb == -999);

assert(convertTune("r//y/g3///o/", instrs, badb) == 0 && instrs == "rxyGGGo" && badb == -999);

assert(convertTune("g3///B/", instrs, badb) == 0 && instrs == "GGGb" && badb == -999); //instrs == GGG/x; changed i iteration -1 > instrs == GGG/b; replace if srarement with else if, removed previous change > solved

assert(convertTune("r/o/y10//////////g/r5/////", instrs, badb) == 0 && instrs == "roYYYYYYYYYYgRRRRR" && badb == -999);

assert(convertTune("r/O/y10//////////g/r5/////", instrs, badb) == 0 && instrs == "roYYYYYYYYYYgRRRRR" && badb == -999);

assert(convertTune("r/", instrs, badb) == 0 && instrs == "r" && badb == -999);

assert(convertTune("r//", instrs, badb) == 0 && instrs == "rx" && badb == -999);

assert(convertTune("//", instrs, badb) == 0 && instrs == "xx" && badb == -999);

assert(convertTune("/", instrs, badb) == 0 && instrs == "x" && badb == -999);

assert(convertTune("", instrs, badb) == 0 && instrs == "" && badb == -999);