report.docx

1. **Notable Obstacles**
   1. The biggest obstacle that I faced arose when I tried to write the code for the encodeSong function. I kept on running into a problem with the octave value being 51 even though the value that I was passing into the function was very obviously within the bounds of the octave. After much grief, I realized that I was passing a character value into the octave instead of an integer. To fix this problem I had to convert the octave character to an octave integer.
      1. For example
         1. int c = 0;

int o = 0;

c = song[1];

o = c-'0';

* 1. Another major obstacle that I faced was figuring out how to correctly place the brackets around the chords. After trying many different types of logic, I eventually found that creating a local variable and temporarily storing parts of the encoded song was the most concise way of adding brackets. To correctly find the chords, a string named temp, temporarily stores notes until the next slash, and an int named instCount kept the count of how many notes were stored in temp. If instCount is greater than 1, then the brackets are added around temp and temp is added to instructions. Lastly temp and instCount are cleared so that the next notes can be checked for chords or not.
  2. The last notable obstacle that I faced was incrementing the for loop correctly for encodeSong, so that all characters in the string were checked. I eventually figured out that after every new substring, I would need to reset the for loop counter to 0. In addition, I would need to replace the old song string with the new substring.
     1. For example
        1. song = song.substr(2, song.size()-2); //new substring

j=0; //resets for loop counter

1. **Design Description (Pseudo Code)**
   1. Int Main
      1. Enter input
      2. Check if hasCorrectSyntax
      3. EncodeSong and return the encoded song
   2. Bool isCorrectChar
      1. Checks for correct letters (A,B,C,D,E,F,G)
      2. Returns true or false
   3. Bool isCorrectAcc
      1. Checks if correct accidentals (#,b)
      2. Returns true or false
   4. Bool isCorrectNum
      1. Checks if correct numbers (2,3,4,5,6)
      2. Returns true or false
   5. Bool hasCorrectSyntax
      1. Check each character of the song string using a loop
         1. To cut out redundancies, parameters checked using the above three functions (isCorrectChar,Num,Acc)
         2. Check if has a char, acc, and num respectively
         3. Check if has char and acc respectively.
         4. Check if has char and num respectively.
         5. Check if has char only.
         6. Check if final char is a slash
            1. If true return true, else false
         7. Else if none of the above statements true return false
      2. Steps ii-vi: if true, cut out the checked portion of song string
   6. encodeSong
      1. Check each character of the song string using loop
         1. Check for C6
         2. Check for Cb6
         3. Check for B#1
         4. Check if has a char, acc, and num respectively
         5. Check if has char and acc respectively. Num auto 4.
         6. Check if has char and num respectively. Acc auto ‘ ‘.
         7. Check if has char only. Num and acc auto 4 and ‘ ‘.
         8. Else return bad beat and 2
         9. Steps 1-7:
            1. Encode note
         10. If double slashes or single slash
             1. If instCount greater than 1

Add brackets

Reset inst count

Clear temp string

* + - * 1. Else

No brackets

Reset instCount

Clear temp string

* + 1. Check if song is playable
       1. Output encoded song and 0
    2. Else return 1
  1. encodeNote
     1. based on octave, char, and acc convert to encoded form.

1. **Test Cases**

|  |  |  |
| --- | --- | --- |
|  | **Song String** | **Reason** |
| **Case1** | “” | Empty returns 0 |
| **Case2** | “C” | Fails, doesn’t have end slash |
| **Case3** | “C#3/” | To check if basic criteria runs |
| **Case4** | "C/C/G/G/A/A/G/" | Return 0 and encode properly |
| **Case5** | “B#1/Cb6/C6/” | Check if special cases run |
| **Case6** | “G3/Fb3/B/F2A0A3/” | Check for return 2 and bb 4 |
| **Case7** | “G2GY//G8//GGG” | Improper syntax |
| **Case8** | “Z/” | Wrong character |
| **Case9** | “A/A/6/” | Wrong syntax |
| **Case10** | “G6/” | Not in playable notes, but proper syntax |
| **Case11** | “C#/” | To check if basic criteria runs |
| **Case12** | “C/” | To check if basic criteria runs |
| **Case13** | “F4G#3/GFA/” | Check for chord brackets |