

16 . MP3.

BLESSON ANDREWS VARGHESE

B110087CS

PROBLEM STATEMENT

Fox Ciel loves music. She currently has n songs in her mp3 player. Their file names are "1.mp3", "2.mp3", and so on, until $\text{string}(n)+\text{"mp3"}$. Sadly, Ciel's mp3 player does not sort the files according to the number. Instead, it simply sorts the file names in lexicographic order, as strings. So, for instance, if $n=10$ then the sorted order looks as follows: "1.mp3", "10.mp3", "2.mp3", ..., "9.mp3". You are given the int n . If n is at most 50, return a `String[]` containing the entire sorted list of file names. If n is more than 50, return a `String[]` containing the first 50 elements of the sorted list of file names.

Definition

Class: `FoxAndMp3`

Method: `playList`

Parameters: `int`

Returns: `String[]`

Method signature: `String[] playList(int n)`

(be sure your method is public)

Notes

- The string A is lexicographically smaller than the string B if either of the following two conditions holds:

1. A is a proper prefix of B ;
2. There is an index i such that the first $(i-1)$ characters of A and B are equal, and character i of A has a smaller ASCII value than character i of B .

Constraints

- n will be between 1 and 1,000,000,000, inclusive.

INPUT SPECIFICATION

The input to the problem will be number of songs present in Fox Ciel's MP3 player. The number of songs must be between 1 and 1000000000. Otherwise program will return an error message and ask user to enter the input again

OUTPUT SPECIFICATION

The output of the problem will be a lexicographically sorted list of the MP3 songs in the player if $n \leq 50$.

The output of the problem will be a lexicographically sorted list of first 50 MP3 songs in the player if $n > 50$.

DESIGN

Program includes 3 header files: `iostream`, `string.h` and vector header files

Program consists of a Class `FoxAndMp3` having a public and private function.

Public function of `FoxAndMp3` is `vector<string> FoxAndMp3::playList(int n)`. It accepts an

integer n representing number of MP3 songs in player and returns a string vector v consisting of lexicographically sorted list of MP3 songs.

Private function of FoxAndMp3 is string FoxAndMp3::num_to_str(int num).It accepts an integer num,convert it into its equivalent string str and returned back.

The main function accepts number of MP3 songs from user and pass it to playList.PlayList stores numbers 1 to n in an array(if n>50, numbers 1 to 50) and sort those lexicographically. Each of these sorted numbers are converted to string using num_to_str and appended to a string vector v using built in function v.push_back.This sting vector v is returned back to main by playList and printed by main() in prescribed format for the user

CORRECTNESS PROOF

Let us prove the correctness of program by explaining correctness of individual functions.

Take an example. Let user input,n=50(worst case)

1.Playlist()

i)It accepts an integer n representing number of MP3 songs in player and return a string vector v consisting of lexicographically sorted list of MP3 songs.

ii)Initially m=5,p=0,q=1,r=0,s=5,t=1,then $11*m-10+p=45$.
since $i \leq 11*m-10+p$ for i in $[0...45]$,for loop will enter ELSE PART OF IF1
(See the comments in program)

iii)For $i=0$,since conditions of IF2 and IF3 are satisfied it will make $a[0]=1$ and $t=2$

iv)For $i=1$ to 8,conditions of IF1 and IF2 are not met and Control transfers to ELSE PART OF IF2 and start executing IF3 statements " $a[i]=q*10 + r$;" and " $r++$;" .Hence number 10 to 18 are stored in $a[1]$ to $a[9]$ and in each step r gets incremented.Now $i=8$,r is 9

v)When $i=10$,as in above step control reaches ELSE PART OF IF2.But at this step $a[9]$ become 19 and r get incremented to 10.Hence condition for IF6 is met and q get incremented to 2 and r get reduced back to 0

vi)Above 3 steps get repeated till value of i is 45.Within this time corresponding values are stored in specific indices of i

$a[11]=2, a[22]=3, a[33]=4, a[44]=5$

$a[12....21]=\{20...29\}$, $a[23...32]=\{30...39\}$, $a[34...43]=\{40...49\}$, $a[45]=50$

vii)When $i=46$,IF1 condition is met and IF1 statements " $a[i]=s+1$;" and " $s++$;" get executed.Hence corresponding values get stored in a.

$a[46]=6$, $a[47]=7$, $a[48]=8$, $a[49]=9$

viii)When $i=50$,control jumps from For loop. All values stored in integer array are converted to corresponding string using function num_to_str and pushed into vector v. v is returned to main

2.num_to_str()

- i)It accepts an integer num,convert it into its equivalent string str and returned back
- ii)Firstly it initialises str to an empty string. It then finds $n = \text{num} \% 10$.
- iii)Since n is found using mod 10,it will be just a single digit. Using a switch we find string corresponding to this digit. This string is appended in beginning of str and n is divided by 10 to eliminate digit we found
- iv)Steps 3 and 4 are repeated till all digits are eliminated and n becomes 0.String ".mp3" is appended to str character by character since we display music files. The string str is returned. It is the corresponding string for argument num

3.main()

- i)The main function accepts number of MP3 songs from user and pass it to PlayList. PlayList returns the Lexicographically sorted list of MP3 files as a string vector whose value is stored in string vector b. Then the contents inside b are displayed using a for loop in the prescribed format for the sake of user. In case the user input is less than 1 or greater than 1000000000 ,System Prints an error Message that "Value Entered is invalid and asks user to enter a number between 1 and 1000000000.

Let user input,n=1(best case)

1.Playlist()

- i)It accepts an integer n representing number of MP3 songs in player and return a string vector v consisting of lexicographically sorted list of MP3 songs.
- ii)Initially $m=0, p=1, q=1, r=0, s=1, t=1$, then $11*m-10+p=-9$. since $i > -9$ for i in $[0...1]$, for loop will enter IF1 (See the comments in program)
- iii)For $i=0$,since conditions of IF1 is satisfied it will make $a[0]=1$ and $s=2$.
- iv)When $i=1$,in next iteration ,control jumps from For loop.All values stored in integer array are converted to corresponding string using function num_to_str and pushed into vector v.v is returned to main

2.num_to_str()

- i)It accepts an integer num,convert it into its equivalent string str and returned back
- ii)Firstly it initialises str to an empty string. It then finds $n = \text{num} \% 10$.
- iii)Since n is found using mod 10,it will be just a single digit. Using a switch we find string corresponding to this digit. This string is appended in beginning of str and n is divided by 10 to eliminate digit we found
- iv)Steps 3 and 4 are repeated till all digits are eliminated and n becomes 0.String

".mp3" is appended to str character by character since we display music files. The string str is returned. It is the corresponding string for argument num

3.main()

i)The main function accepts number of MP3 songs from user and pass it to playList. PlayList returns the Lexicographically sorted list of MP3 files as a string vector whose value is stored in string vector b. Then the contents inside b are displayed using a for loop in the prescribed format for the sake of user. In case the user input is less than 1 or greater than 1000000000 ,System Prints an error Message that "Value Entered is invalid and asks user to enter a number b/w 1 and 1000000000.

Let user input,n=25(average case)

1.Playlist()

i)It accepts an integer n representing number of MP3 songs in player and return a string vector v consisting of lexicographically sorted list of MP3 songs.

ii)Initially $m=2, p=5, q=1, r=0, s=2, t=1$, then $11*m-10+p=17$.
since $i \leq 11*m-10+p$ for i in $[0...17]$, for loop will enter ELSE PART OF IF1
(See the comments in program)

iii)For $i=0$, since conditions of IF2 and IF3 are satisfied it will make $a[0]=1$ and $t=2$

iv)For $i=1$ to 8, conditions of IF1 and IF2 are not met and Control transfers to ELSE PART OF IF2 and start executing IF3 statements " $a[i]=q*10 + r$;" and " $r++$;" .Hence number 10 to 18 are stored in $a[1]$ to $a[9]$ and in each step r gets incremented. Now $i=8, r$ is 9

v)When $i=10$, as in above step control reaches ELSE PART OF IF2. But at this step $a[9]$ become 19 and r get incremented to 10. Hence condition for IF6 is met and q get incremented to 2 and r get reduced back to 0

vi)Above 3 steps get repeated till value of i is 17. Within this time corresponding values are stored in specific indices of i
 $a[11]=2$,
 $a[12...17]=\{20...25\}$,

vii)When $i=17$, IF1 condition is met and IF1 statements " $a[i]=s+1$;" and " $s++$;" get executed .Hence corresponding values get stored in a .
 $a[18]=3$, $a[19]=4$, $a[20]=5$, $a[21]=6$, $a[22]=7$, $a[23] =8$, $a[24] =9$

viii)When $i=25$, control jumps from For loop. All values stored in integer array are converted to corresponding string using function num_to_str and pushed into vector v. v is returned to main

2.num_to_str()

i) It accepts an integer num, convert it into its equivalent string str and returned back

ii) Firstly it initialises str to an empty string. It then finds $n = \text{num} \% 10$.

iii) Since n is found using mod 10, it will be just a single digit. Using a switch we find string corresponding to this digit. This string is appended in beginning of `str` and n is divided by 10 to eliminate digit we found

iv) Steps 3 and 4 are repeated till all digits are eliminated and n becomes 0. String ".mp3" is appended to str character by character since we display music files. The string str is returned. It is the corresponding string for argument num

3.main()

i) The main function accepts number of MP3 songs from user and pass it to playList. PlayList returns the Lexicographically sorted list of MP3 files as a string vector whose value is stored in string vector b. Then the contents inside b are displayed using a for loop in the prescribed format for the sake of user. In case the user input is less than 1 or greater than 1000000000, System Prints an error Message that "Value Entered is invalid and asks user to enter a number between 1 and 1000000000.

HENCE ,we get correct outputs for the worst case , best case and any average case .Therefore the program works perfectly.

In case user enters any number less than 1 or greater than 1000000000. Program will print an error message in the main function itself.

Hence Correctness of the code is proved..

SAMPLE INPUTS

(i) 1

(ii) 3

(iii) 16

(iv) 50

(v) 51

(vi) 10000000000000000000000000000000000

(vii) 0

SAMPLE OUTPUTS

(i)

~~~SONGS LIST~~~

{1.mp3  
}

(ii)

{1.mp3  
2.mp3  
3.mp3  
}

(iii)

~~~SONGS LIST~~~

{1.mp3
10.mp3
11.mp3
12.mp3
13.mp3
14.mp3
15.mp3
16.mp3
2.mp3
3.mp3
4.mp3
5.mp3
6.mp3
7.mp3
8.mp3
9.mp3
}

(iv)

~~~SONGS LIST~~~

{1.mp3  
10.mp3  
11.mp3  
12.mp3  
13.mp3  
14.mp3  
15.mp3  
16.mp3  
17.mp3  
18.mp3  
19.mp3  
2.mp3

20.mp3  
21.mp3  
22.mp3  
23.mp3  
24.mp3  
25.mp3  
26.mp3  
27.mp3  
28.mp3  
29.mp3  
3.mp3  
30.mp3  
31.mp3  
32.mp3  
33.mp3  
34.mp3  
35.mp3  
36.mp3  
37.mp3  
38.mp3  
39.mp3  
4.mp3  
40.mp3  
41.mp3  
42.mp3  
43.mp3  
44.mp3  
45.mp3  
46.mp3  
47.mp3  
48.mp3  
49.mp3  
5.mp3  
50.mp3  
6.mp3  
7.mp3  
8.mp3  
9.mp3  
}

(v)

First 50 songs are

{1.mp3  
10.mp3  
11.mp3  
12.mp3  
13.mp3  
14.mp3  
15.mp3

16.mp3  
17.mp3  
18.mp3  
19.mp3  
2.mp3  
20.mp3  
21.mp3  
22.mp3  
23.mp3  
24.mp3  
25.mp3  
26.mp3  
27.mp3  
28.mp3  
29.mp3  
3.mp3  
30.mp3  
31.mp3  
32.mp3  
33.mp3  
34.mp3  
35.mp3  
36.mp3  
37.mp3  
38.mp3  
39.mp3  
4.mp3  
40.mp3  
41.mp3  
42.mp3  
43.mp3  
44.mp3  
45.mp3  
46.mp3  
47.mp3  
48.mp3  
49.mp3  
5.mp3  
50.mp3  
6.mp3  
7.mp3  
8.mp3  
9.mp3  
}

(vi)

Value Entered is invalid. Please enter a number b/w 1 and 1000000000  
enter the number of songs

25 /\* USER INPUT \*/\



~~~SONGS LIST~~~

{1.mp3
10.mp3
11.mp3
12.mp3
13.mp3
14.mp3
15.mp3
16.mp3
17.mp3
18.mp3
19.mp3
2.mp3
20.mp3
21.mp3
22.mp3
23.mp3
24.mp3
25.mp3
3.mp3
4.mp3
5.mp3
6.mp3
7.mp3
8.mp3
9.mp3
}

(vii)

Value Entered is invalid.Please enter a number b/w 1 and 1000000000

enter the number of songs

20 /* USER INPUT *\

~~~SONGS LIST~~~

{1.mp3  
10.mp3  
11.mp3  
12.mp3  
13.mp3  
14.mp3  
15.mp3  
16.mp3  
17.mp3  
18.mp3  
19.mp3  
2.mp3  
20.mp3  
3.mp3

```
4.mp3
5.mp3
6.mp3
7.mp3
8.mp3
9.mp3
}
```

## SCREENSHOTS

Screen shot-1

[illegible]

Screen shot-2

```
Applications Places System VCD 094 USA 11:57:41 AM blessonav
blessonav@blessonav-Studio-1569: ~/myxos1/B110087CS_MP3_blesson
File Edit View Terminal Help
blessonav@blessonav-Studio-1569:~/myxos1/B110087CS_MP3_blesson$ ./a.out
enter the number of songs
25

~~~SONGS LIST~~~
{1.mp3
10.mp3
11.mp3
12.mp3
13.mp3
14.mp3
15.mp3
16.mp3
17.mp3
18.mp3
19.mp3
2.mp3
20.mp3
21.mp3
22.mp3
23.mp3
24.mp3
25.mp3
3.mp3
4.mp3
5.mp3
6.mp3
7.mp3
8.mp3
9.mp3
}

blessonav@blessonav-Studio-1569:~/myxos1/B110087CS_MP3_blesson$./a.out
enter the number of songs
50

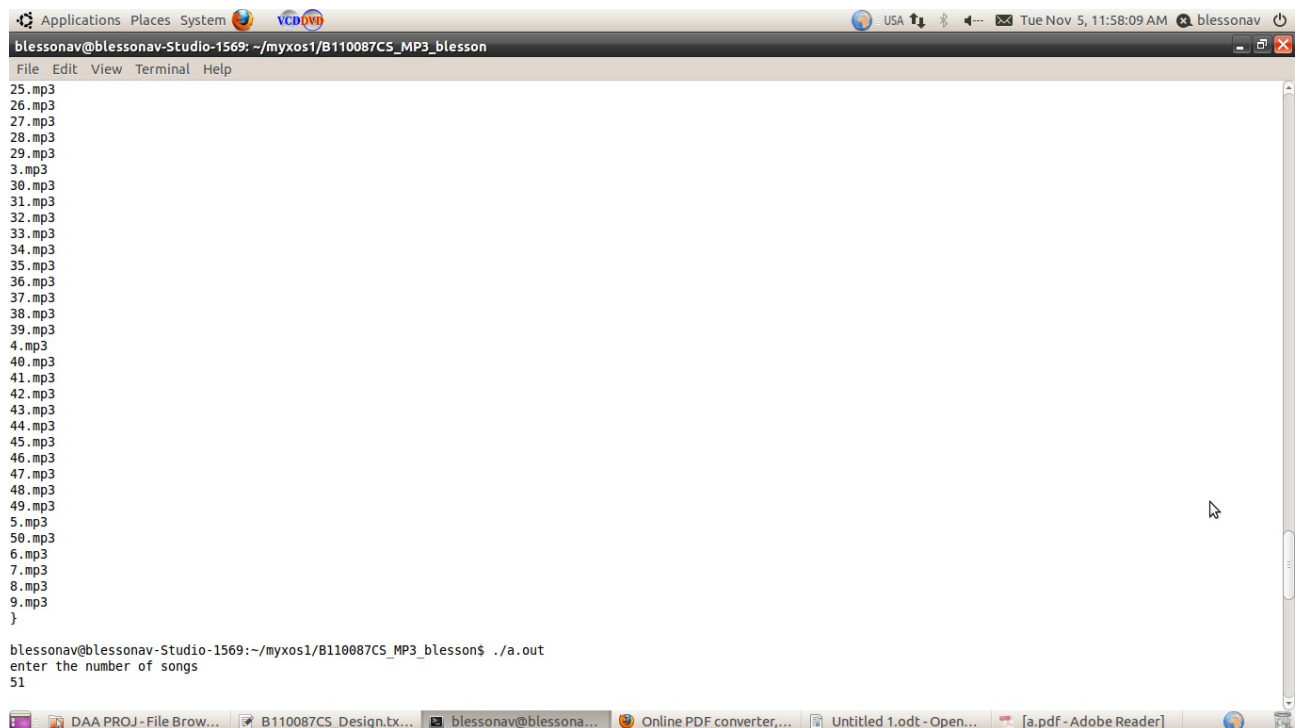
~~~SONGS LIST~~~
```

### Screen shot-3

```
Applications Places System VCD 094 USA 11:57:55 AM blessonav
blessonav@blessonav-Studio-1569: ~/myxos1/B110087CS_MP3_blesson
File Edit View Terminal Help
blessonav@blessonav-Studio-1569:~/myxos1/B110087CS_MP3_blesson$ ./a.out
enter the number of songs
50

~~~SONGS LIST~~~
{1.mp3
10.mp3
11.mp3
12.mp3
13.mp3
14.mp3
15.mp3
16.mp3
17.mp3
18.mp3
19.mp3
2.mp3
20.mp3
21.mp3
22.mp3
23.mp3
24.mp3
25.mp3
26.mp3
27.mp3
28.mp3
29.mp3
3.mp3
30.mp3
31.mp3
32.mp3
33.mp3
34.mp3
35.mp3
36.mp3
37.mp3
38.mp3
39.mp3
}
```

## Screen shot-4

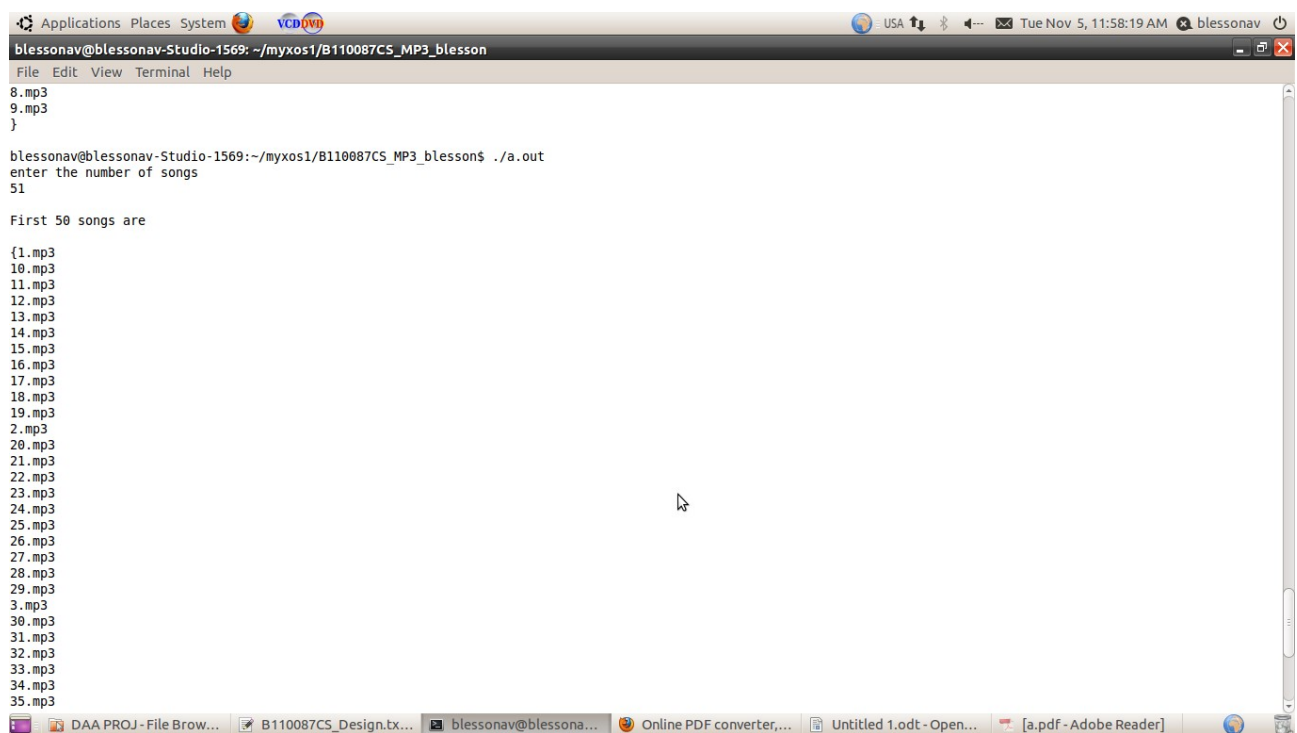


A screenshot of a terminal window titled "blessonav@blessonav-Studio-1569: ~/myxos1/B110087CS\_MP3\_blesson". The terminal displays a list of 51 MP3 files, numbered 1 through 51, arranged in a specific order. The list is as follows:

- 25.mp3
- 26.mp3
- 27.mp3
- 28.mp3
- 29.mp3
- 3.mp3
- 30.mp3
- 31.mp3
- 32.mp3
- 33.mp3
- 34.mp3
- 35.mp3
- 36.mp3
- 37.mp3
- 38.mp3
- 39.mp3
- 4.mp3
- 40.mp3
- 41.mp3
- 42.mp3
- 43.mp3
- 44.mp3
- 45.mp3
- 46.mp3
- 47.mp3
- 48.mp3
- 49.mp3
- 5.mp3
- 50.mp3
- 6.mp3
- 7.mp3
- 8.mp3
- 9.mp3

The prompt "blessonav@blessonav-Studio-1569:~/myxos1/B110087CS\_MP3\_blesson\$ ./a.out" is shown, followed by the input "enter the number of songs" and the output "51". The terminal window has a menu bar with "File", "Edit", "View", "Terminal", and "Help". The system tray at the bottom shows various icons, including "DAA PROJ - File Brow...", "B110087CS\_Design.tx...", "blessonav@blessona...", "Online PDF converter...", "Untitled 1.odt - Open...", and "[a.pdf - Adobe Reader]".

## Screen shot -5

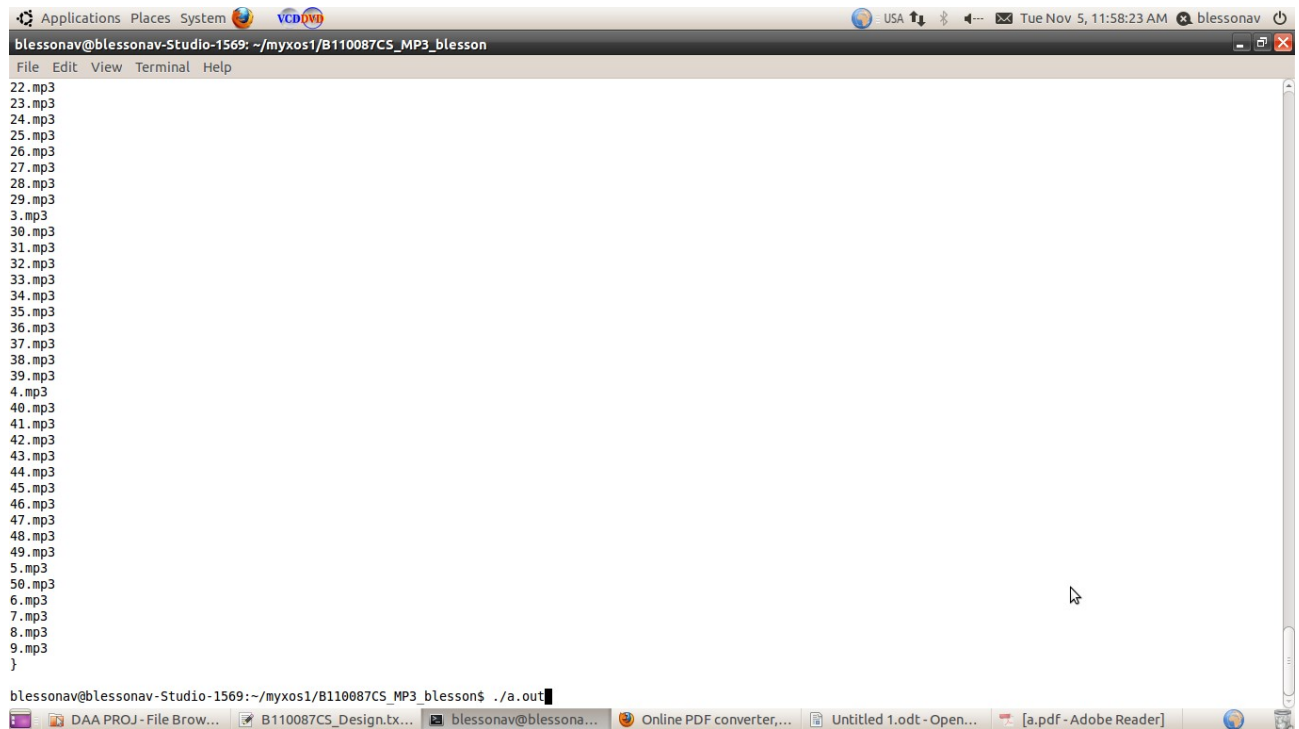


A screenshot of a terminal window titled "blessonav@blessonav-Studio-1569: ~/myxos1/B110087CS\_MP3\_blesson". The terminal displays a list of 51 MP3 files, numbered 1 through 51, arranged in a specific order. The list is as follows:

- 8.mp3
- 9.mp3
- 10.mp3
- 11.mp3
- 12.mp3
- 13.mp3
- 14.mp3
- 15.mp3
- 16.mp3
- 17.mp3
- 18.mp3
- 19.mp3
- 20.mp3
- 21.mp3
- 22.mp3
- 23.mp3
- 24.mp3
- 25.mp3
- 26.mp3
- 27.mp3
- 28.mp3
- 29.mp3
- 30.mp3
- 31.mp3
- 32.mp3
- 33.mp3
- 34.mp3
- 35.mp3
- 36.mp3
- 37.mp3
- 38.mp3
- 39.mp3
- 40.mp3
- 41.mp3
- 42.mp3
- 43.mp3
- 44.mp3
- 45.mp3
- 46.mp3
- 47.mp3
- 48.mp3
- 49.mp3
- 50.mp3
- 51.mp3

The prompt "blessonav@blessonav-Studio-1569:~/myxos1/B110087CS\_MP3\_blesson\$ ./a.out" is shown, followed by the input "enter the number of songs" and the output "51". The terminal window has a menu bar with "File", "Edit", "View", "Terminal", and "Help". The system tray at the bottom shows various icons, including "DAA PROJ - File Brow...", "B110087CS\_Design.tx...", "blessonav@blessona...", "Online PDF converter...", "Untitled 1.odt - Open...", and "[a.pdf - Adobe Reader]".

## Screen shot-7



The screenshot shows a Linux terminal window with the title bar "Applications Places System" and "VCD OVI". The terminal prompt is "blessonav@blessonav-Studio-1569: ~/myxos1/B110087CS\_MP3\_blesson". The terminal content displays a list of MP3 files, including "22.mp3" through "39.mp3", "4.mp3", "40.mp3" through "49.mp3", "5.mp3", "50.mp3", "6.mp3", "7.mp3", "8.mp3", "9.mp3", and "}". The terminal prompt is "blessonav@blessonav-Studio-1569:~/myxos1/B110087CS\_MP3\_blesson\$ ./a.out". The terminal window is part of a desktop environment with a taskbar at the bottom showing various applications like "DAA PROJ - File Brow...", "B110087CS\_Design.tx...", "blessonav@blessona...", "Online PDF converter,...", "Untitled 1.odt - Open...", and "[a.pdf - Adobe Reader]".

```
blessonav@blessonav-Studio-1569:~/myxos1/B110087CS_MP3_blesson$./a.out
```

END