

Terminal Simulation

locked

Problem

Submissions

Leaderboard

Discussions

In this challenge, the goal is to simulate the behavior of 4 different unix terminals: sh, csh, bash, zsh. These terminals provide commands to navigate the file system, and for creation and deletion of files and folders. However, there are some minor differences among all 4 terminals. First, we list the common functionality that needs to be implemented as part of the challenge.

Common Functionality

- `pwd` : Prints the current directory.
- `mkdir <dirname>` : Creates a new (empty) directory named `<dirname>` inside the current directory.
- `touch <filename>` : Creates a new file named `<filename>` inside the current directory.
- `rm <name>` : Remove the file/folder named `<name>` from the current directory. If a folder is being removed, it will also remove everything inside the folder.
- `cd <dirname>` : Change the current directory to `<dirname>` if it exists in the current directory.
- `ls` : Print the contents of the current directory. All folders should be printed first in alphabetical order, followed by all files in alphabetical order (everything is space separated). You can check the sample test cases for the exact format. Note that there should be one more space after the last entry.
- `quit` : Quits the terminal session.

All file names and folder names are specified by the regular expression: $([a-z][A-Z][0-9]_)^+$ (except the special cases for `cd` as mentioned below). This is only for your information, you don't have to verify this as part of your implementation.

Error Messages

- For `mkdir` and `touch`, if the specified folder/file already exists (in the current directory), print `Folder Exists` or `File Exists` respectively. Note that there cannot be a folder and file with the same name in the same directory.
- For `cd`, if the specified directory does not exist in the current directory, print `Folder does not exist`.
- For `rm`, if the specified folder/file does not exist, print `Does not exist`.
- Note that `pwd` and `ls` are the only commands which will print anything in a normal run.
- If an unknown command is given as input, print `Command does not exist`.

Directory structure

- The root directory is `/`. At the beginning, the current directory will be the home directory of the logged in user, given by `/home/<username>`.
- `cd ..` will change current directory to parent directory. If this command is invoked when the current directory is the root directory, the current directory will remain unchanged.
- Unlike real-world terminals, all commands in this challenge only take as argument a file/folder name, and not the complete path. You can assume that `/` will never appear in the arguments of any command.

Different functionality across terminals

- **sh** implements all the above functionality, and uses `$` as a prompt.
- **csh** implements all functionality of **sh**, uses `$` as a prompt, and in addition also allows a special argument `~` to `cd`, which changes the current directory to the home directory of the logged in user. For instance, if user `alice` is logged in, then `cd ~` will change the current directory to `/home/alice`
- **bash** implements all functionality of **csh**, but uses `<currDirName> $` as a prompt, where `<currDirName>` is the current directory name.
- **zsh** implements all functionality of **bash**, but uses `<userName> <currDirName> $` as a prompt, where `<userName>` is the logged user. In addition, it also allows a command called `history` which prints all the commands executed so far in a chronological order with numbering starting from 0. It should print all executed commands including the invalid ones.

It is **highly recommended** to use various OO features covered during the course in your implementation. In particular, features like inheritance, overloading, exceptions may come in very handy. However, it is **not compulsory** to use any OO feature. There are **no complexity constraints** and **no restrictions** on use of STL data structures and functions.

Input Format

- The first line will be `<terminalName> <username>` where `<TerminalName>` will be one of `sh,csh,bash,zsh`. You need to start the corresponding terminal session, with the current directory being `/home/<username>`.
- Next will be a series of commands and arguments, with 1 command per line. The last command will be `quit`. You can assume that if a command requires an argument, it will be given.

Constraints

None.

Output Format

You need to print both the input and output for the entire terminal session. Depending on which terminal is called, use the appropriate prompt, print the input command, and then its output (if any). That is, suppose the first command is `<commandName>` `<arg>` and the terminal is `bash`, then you need to print:

```
<currDirname> $ <commandName> <arg>
<output>
```

Note the space before and after `$`. Do this for every command in the input, until you reach `quit`. See the sample testcases below for a better idea.

Sample Input 0

```
sh naruto
mkdir folder1
mkdir folder2
mkdir folder5
mkdir folder4
mkdir folder3
ls
pwd
quit
```

Sample Output 0

```
$ mkdir folder1
$ mkdir folder2
$ mkdir folder5
$ mkdir folder4
$ mkdir folder3
$ ls
folder1 folder2 folder3 folder4 folder5
```

```
$ pwd
/home/naruto
$ quit
```

Sample Input 1

```
csh gara
mkdir folder2
ls
touch file2
ls
touch file1
ls
pwd
quit
```

Sample Output 1

```
$ mkdir folder2
$ ls
folder2
$ touch file2
$ ls
folder2 file2
$ touch file1
$ ls
folder2 file1 file2
$ pwd
/home/gara
$ quit
```

Sample Input 2

```
bash neji
mkdir folder3
mkdir folder2
mkdir folder1
ls
touch file3
ls
pwd
rm folder3
ls
rm file3
ls
quit
```

Sample Output 2

```
/home/neji $ mkdir folder3
/home/neji $ mkdir folder2
/home/neji $ mkdir folder1
/home/neji $ ls
folder1 folder2 folder3
/home/neji $ touch file3
/home/neji $ ls
folder1 folder2 folder3 file3
/home/neji $ pwd
/home/neji
/home/neji $ rm folder3
/home/neji $ ls
folder1 folder2 file3
/home/neji $ rm file3
/home/neji $ ls
folder1 folder2
/home/neji $ quit
```

Sample Input 3

```
sh L_lalit
pwd
touch file2
touch file1
ls
mkdir l
cd l
mkdir a
cd a
mkdir l
cd l
mkdir i
cd i
mkdir t
cd t
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
cd ..
pwd
ls
quit
```

Sample Output 3

```
$ pwd
/home/L_lalit
$ touch file2
$ touch file1
$ ls
file1 file2
$ mkdir l
$ cd l
$ mkdir a
$ cd a
$ mkdir l
$ cd l
$ mkdir i
$ cd i
$ mkdir t
$ cd t
$ pwd
/home/L_lalit/l/a/l/i/t
$ cd ..
$ pwd
/home/L_lalit/l/a/l/i
$ cd ..
$ pwd
/home/L_lalit/l/a/l
$ cd ..
$ pwd
/home/L_lalit/l/a
$ cd ..
$ pwd
/home/L_lalit/l
$ cd ..
$ pwd
/home/L_lalit
```

```
$ cd ..  
$ pwd  
/home  
$ cd ..  
$ pwd  
/  
$ cd ..  
$ pwd  
/  
$ ls  
home  
$ quit
```

Sample Input 4

```
ssh bob  
pwd  
mkdir Pictures  
touch file1  
ls  
cd Pictures  
mkdir Beach  
cd Beach  
touch file4  
ls  
pwd  
cd ~  
ls  
pwd  
quit
```

Sample Output 4

```
$ pwd  
/home/bob  
$ mkdir Pictures  
$ touch file1  
$ ls  
Pictures file1  
$ cd Pictures  
$ mkdir Beach  
$ cd Beach  
$ touch file4  
$ ls  
file4  
$ pwd  
/home/bob/Pictures/Beach  
$ cd ~  
$ ls  
Pictures file1  
$ pwd  
/home/bob  
$ quit
```

Sample Input 5

```
sh alice  
mkdir folder1  
touch file1  
ls  
cd folder1  
pwd  
history  
cd ..  
pwd  
quit
```

Sample Output 5

```
$ mkdir folder1
$ touch file1
$ ls
folder1 file1
$ cd folder1
$ pwd
/home/alice/folder1
$ history
Command does not exist
$ cd ..
$ pwd
/home/alice
$ quit
```

Sample Input 6

```
csh alice
mkdir folder1
ls
touch file2
ls
cd folder1
pwd
cd ~
pwd
rm folder3
rm folder1
ls
quit
```

Sample Output 6

```
$ mkdir folder1
$ ls
folder1
$ touch file2
$ ls
folder1 file2
$ cd folder1
$ pwd
/home/alice/folder1
$ cd ~
$ pwd
/home/alice
$ rm folder3
Does not exist
$ rm folder1
$ ls
file2
$ quit
```

Sample Input 7

```
sh tan
mkdir doc1
ls
mkdir doc1
touch file1
ls
touch file1
cd doc1
pwd
cd ..
rm doc1
ls
quit
```

Sample Output 7

```
$ mkdir doc1
$ ls
doc1
$ mkdir doc1
Folder Exists
$ touch file1
$ ls
doc1 file1
$ touch file1
File Exists
$ cd doc1
$ pwd
/home/tan/doc1
$ cd ..
$ rm doc1
$ ls
file1
$ quit
```

Sample Input 8

```
bash alice
mkdir folder3
ls
touch file3
ls
cd folder3
pwd
cd ~
rm folder3
ls
cd folder3
quit
```

Sample Output 8

```
/home/alice $ mkdir folder3
/home/alice $ ls
folder3
/home/alice $ touch file3
/home/alice $ ls
folder3 file3
/home/alice $ cd folder3
/home/alice/folder3 $ pwd
/home/alice/folder3
/home/alice/folder3 $ cd ~
/home/alice $ rm folder3
/home/alice $ ls
file3
/home/alice $ cd folder3
Folder does not exist
/home/alice $ quit
```

Sample Input 9

```
zsh alice
mkdir folder4
ls
touch file4
ls
cd folder4
pwd
cd ..
rm folder4
ls
pwd
```

```
cd folder4
history
quit
```

Sample Output 9

```
alice /home/alice $ mkdir folder4
alice /home/alice $ ls
folder4
alice /home/alice $ touch file4
alice /home/alice $ ls
folder4 file4
alice /home/alice $ cd folder4
alice /home/alice/folder4 $ pwd
/home/alice/folder4
alice /home/alice/folder4 $ cd ..
alice /home/alice $ rm folder4
alice /home/alice $ ls
file4
alice /home/alice $ pwd
/home/alice
alice /home/alice $ cd folder4
Folder does not exist
alice /home/alice $ history
0 mkdir folder4
1 ls
2 touch file4
3 ls
4 cd folder4
5 pwd
6 cd ..
7 rm folder4
8 ls
9 pwd
10 cd folder4
alice /home/alice $ quit
```

[f](#) [t](#) [in](#)

Submissions: 83

Max Score: 100

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

C++20



```
1 #include <cmath>
2 #include <cstdio>
3 #include <stack>
4 #include <vector>
5 #include <iostream>
6 #include <algorithm>
7 using namespace std;
8 struct Drc
9 {
10     string s;
11     Drc* h;
12     bool b;
13     vector<Drc*> vec1;
14 };
15 class Directory_sh
16 {
17     public:
```



```
18  Drc* home;
19  string start;
20  Drc* root;
21  Drc* current;
22  Directory_sh()
23  {
24      home = new Drc;
25      home->s = "home";
26      string g;
27      cin>>g;
28      start = g;
29      Drc* child = new Drc;
30      child->s = g;
31      child->b = true;
32      (home->vec1).push_back(child);
33      root = new Drc;
34      root->s = "";
35      root->b = true;
36      (root->vec1).push_back(home);
37      child->h = home;
38      home->h = root;
39      current = child;
40  }
41  void pwd()
42  {
43      cout<<"$ pwd"<<endl;
44      Drc* dd = current;
45      if (current==root)
46      {
47          cout<<"/"<<endl;
48          return;
49      }
50      stack<string> st;
51      while(dd!=root)
52      {
53          st.push(dd->s);
54          dd = dd->h;
55      }
56      while(!st.empty())
57      {
58          cout<<"/"<<st.top();
59          st.pop();
60      }
61      cout<<endl;
62  }
63  void mkdir(string str)
64  {
65      cout<<"$ mkdir"<<" "<<str<<endl;
66      for (auto z : current->vec1)
67      {
68          if (z->s == str)
69          {
70              if (z->b==true)
71              {
72                  cout<<"Folder Exists"<<endl;
73                  return;
74              }
75              else
76              {
77                  cout<<"File Exists"<<endl;
78                  return;
79              }
80          }
81      }
82      Drc* child = new Drc;
83      child->s = str;
```

```
84     child->b = true;
85     child->h = current;
86     (current->vec1).push_back(child);
87 }
88 void touch(string str)
89 {
90     cout<<"$ touch"<<" "<<str<<endl;
91     for (auto z : current->vec1)
92     {
93         if (z->s == str)
94         {
95             if (z->b==true)
96             {
97                 cout<<"Folder Exists"<<endl;
98                 return;
99             }
100             else
101             {
102                 cout<<"File Exists"<<endl;
103                 return;
104             }
105         }
106     }
107     Drc* child = new Drc;
108     child->s = str;
109     child->b = false;
110     child->h = current;
111     (current->vec1).push_back(child);
112 }
113 void rm(string str)
114 {
115     cout<<"$ rm "<<str<<endl;
116     int ii=0;
117     for (auto z : current->vec1)
118     {
119         if (z->s == str)
120         {
121             (current->vec1).erase(current->vec1.begin() + ii);
122             return;
123         }
124         ii++;
125     }
126     cout<<"Does not exist"<<endl;
127 }
128 void cd(string str)
129 {
130     cout<<"$ cd "<<str<<endl;
131     if (str=="..")
132     {
133         if (current!=root)
134         {
135             current = current->h;
136         }
137         return;
138     }
139     for (auto z : current->vec1)
140     {
141         if (z->s==str && z->b==true)
142         {
143             current = z;
144             return;
145         }
146     }
147     cout<<"Folder does not exist"<<endl;
148 }
149 void ls()
```

```
150 {
151     cout<<"$ ls"<<endl;
152     vector<string> folder;
153     vector<string> file;
154     for (auto z : current->vec1)
155     {
156         if (z->b==true)
157         {
158             folder.push_back(z->s);
159         }
160         else
161         {
162             file.push_back(z->s);
163         }
164     }
165     sort(folder.begin(), folder.end());
166     sort(file.begin(), file.end());
167     for (auto z : folder)
168     {
169         cout<<z<<" ";
170     }
171     for (auto z : file)
172     {
173         cout<<z<<" ";
174     }
175     cout<<endl;
176 }
177 void quit()
178 {
179     cout<<"$ quit"<<endl;
180 }
181 };
182 class Directory_csh
183 {
184     public:
185     Drc* home;
186     string start;
187     Drc* root;
188     Drc* current;
189     Directory_csh()
190     {
191         home = new Drc;
192         home->s = "home";
193         string g;
194         cin>>g;
195         start = g;
196         Drc* child = new Drc;
197         child->s = g;
198         child->b = true;
199         (home->vec1).push_back(child);
200         root = new Drc;
201         root->s = "";
202         root->b = true;
203         (root->vec1).push_back(home);
204         child->h = home;
205         home->h = root;
206         current = child;
207     }
208     void pwd()
209     {
210         cout<<"$ pwd"<<endl;
211         Drc* dd = current;
212         if (current==root)
213         {
214             cout<<"/"<<endl;
215             return;
```

```
216     }
217     stack<string> st;
218     while(dd!=root)
219     {
220         st.push(dd->s);
221         dd = dd->h;
222     }
223     while(!st.empty())
224     {
225         cout<<"/"<<st.top();
226         st.pop();
227     }
228     cout<<endl;
229 }
230 void mkdir(string str)
231 {
232     cout<<"$ mkdir"<<" "<<str<<endl;
233     for (auto z : current->vec1)
234     {
235         if (z->s == str)
236         {
237             if (z->b==true)
238             {
239                 cout<<"Folder Exists"<<endl;
240                 return;
241             }
242             else
243             {
244                 cout<<"File Exists"<<endl;
245                 return;
246             }
247         }
248     }
249     Drc* child = new Drc;
250     child->s = str;
251     child->b = true;
252     child->h = current;
253     (current->vec1).push_back(child);
254 }
255 void touch(string str)
256 {
257     cout<<"$ touch"<<" "<<str<<endl;
258     for (auto z : current->vec1)
259     {
260         if (z->s == str)
261         {
262             if (z->b==true)
263             {
264                 cout<<"Folder Exists"<<endl;
265                 return;
266             }
267             else
268             {
269                 cout<<"File Exists"<<endl;
270                 return;
271             }
272         }
273     }
274     Drc* child = new Drc;
275     child->s = str;
276     child->b = false;
277     child->h = current;
278     (current->vec1).push_back(child);
279 }
280 void rm(string str)
281 {
```

```
282     cout<<"$ rm "<<str<<endl;
283     int ii=0;
284     for (auto z : current->vec1)
285     {
286         if (z->s == str)
287         {
288             (current->vec1).erase(current->vec1.begin() + ii);
289             return;
290         }
291         ii++;
292     }
293     cout<<"Does not exist"<<endl;
294 }
295 void cd(string str)
296 {
297     cout<<"$ cd "<<str<<endl;
298     if (str=="..")
299     {
300         if (current!=root)
301         {
302             current = current->h;
303         }
304         return;
305     }
306     else if (str=="~")
307     {
308         for (auto z : home->vec1)
309         {
310             if (z->s == start)
311             {
312                 current = z;
313             }
314         }
315         //current = home->vec1[0];
316         return;
317     }
318     for (auto z : current->vec1)
319     {
320         if (z->s==str && z->b==true)
321         {
322             current = z;
323             return;
324         }
325     }
326     cout<<"Folder does not exist"<<endl;
327 }
328 void ls()
329 {
330     cout<<"$ ls"<<endl;
331     vector<string> folder;
332     vector<string> file;
333     for (auto z : current->vec1)
334     {
335         if (z->b==true)
336         {
337             folder.push_back(z->s);
338         }
339         else
340         {
341             file.push_back(z->s);
342         }
343     }
344     sort(folder.begin(), folder.end());
345     sort(file.begin(), file.end());
346     for (auto z : folder)
347     {
```

```
348         cout<<z<<" ";
349     }
350     for (auto z : file)
351     {
352         cout<<z<<" ";
353     }
354     cout<<endl;
355 }
356 void quit()
357 {
358     cout<<"$ quit"<<endl;
359 }
360 };
361 class Directory_bash
362 {
363     public:
364     Drc* home;
365     string start;
366     Drc* root;
367     Drc* current;
368     Directory_bash()
369     {
370         home = new Drc;
371         home->s = "home";
372         string g;
373         cin>>g;
374         start = g;
375         Drc* child = new Drc;
376         child->s = g;
377         child->b = true;
378         (home->vec1).push_back(child);
379         root = new Drc;
380         root->s = "";
381         root->b = true;
382         (root->vec1).push_back(home);
383         child->h = home;
384         home->h = root;
385         current = child;
386     }
387     void pwd()
388     {
389         Drc* dd = current;
390         stack<string> st;
391         if (current==root)
392         {
393             cout<<"/"<<endl;
394             //return;
395         }
396         else
397         {
398             while(dd!=root)
399             {
400                 st.push(dd->s);
401                 dd = dd->h;
402             }
403             while(!st.empty())
404             {
405                 cout<<"/"<<st.top();
406                 st.pop();
407             }
408             cout<<" $ pwd"<<endl;
409             dd = current;
410             if (current==root)
411             {
412                 cout<<"/"<<endl;
413             }
```

```
414         return;
415     }
416     while(dd!=root)
417     {
418         st.push(dd->s);
419         dd = dd->h;
420     }
421     while(!st.empty())
422     {
423         cout<<"/"<<st.top();
424         st.pop();
425     }
426     cout<<endl;
427 }
428 void mkdir(string str)
429 {
430     Drc* dd = current;
431     if (current==root)
432     {
433         cout<<"/"<<endl;
434         //return;
435     }
436     else
437     {
438         stack<string> st;
439         while(dd!=root)
440         {
441             st.push(dd->s);
442             dd = dd->h;
443         }
444         while(!st.empty())
445         {
446             cout<<"/"<<st.top();
447             st.pop();
448         }
449     }
450     cout<<" $ mkdir"<<" "<<str<<endl;
451     for (auto z : current->vec1)
452     {
453         if (z->s == str)
454         {
455             if (z->b==true)
456             {
457                 cout<<"Folder Exists"<<endl;
458                 return;
459             }
460             else
461             {
462                 cout<<"File Exists"<<endl;
463                 return;
464             }
465         }
466     }
467     Drc* child = new Drc;
468     child->s = str;
469     child->b = true;
470     child->h = current;
471     (current->vec1).push_back(child);
472 }
473 void touch(string str)
474 {
475     Drc* dd = current;
476     if (current==root)
477     {
478         cout<<"/"<<endl;
479         //return;
```

```
480     }
481     {
482     stack<string> st;
483     while(dd!=root)
484     {
485         st.push(dd->s);
486         dd = dd->h;
487     }
488     while(!st.empty())
489     {
490         cout<<"/"<<st.top();
491         st.pop();
492     }
493     }
494     cout<<" $ touch"<<" "<<str<<endl;
495     for (auto z : current->vec1)
496     {
497         if (z->s == str)
498         {
499             if (z->b==true)
500             {
501                 cout<<"Folder Exists"<<endl;
502                 return;
503             }
504             else
505             {
506                 cout<<"File Exists"<<endl;
507                 return;
508             }
509         }
510     }
511     Drc* child = new Drc;
512     child->s = str;
513     child->b = false;
514     child->h = current;
515     (current->vec1).push_back(child);
516 }
517 void rm(string str)
518 {
519     Drc* dd = current;
520     if (current==root)
521     {
522         cout<<"/"<<endl;
523         //return;
524     }
525     else
526     {
527         stack<string> st;
528         while(dd!=root)
529         {
530             st.push(dd->s);
531             dd = dd->h;
532         }
533         while(!st.empty())
534         {
535             cout<<"/"<<st.top();
536             st.pop();
537         }
538     }
539     cout<<" $ rm "<<str<<endl;
540     int ii=0;
541     for (auto z : current->vec1)
542     {
543         if (z->s == str)
544         {
545             (current->vec1).erase(current->vec1.begin() + ii);
```



```
546         return;
547     }
548     ii++;
549 }
550 cout<<"Does not exist"<<endl;
551 }
552 void cd(string str)
553 {
554     Drc* dd = current;
555     if (current==root)
556     {
557         cout<<"/"<<endl;
558         //return;
559     }
560     else
561     {
562         stack<string> st;
563         while(dd!=root)
564         {
565             st.push(dd->s);
566             dd = dd->h;
567         }
568         while(!st.empty())
569         {
570             cout<<"/"<<st.top();
571             st.pop();
572         }
573     }
574     cout<<" $ cd " <<str<<endl;
575     if (str=="..")
576     {
577         if (current!=root)
578         {
579             current = current->h;
580         }
581         return;
582     }
583     else if (str=="~")
584     {
585         for (auto z : home->vec1)
586         {
587             if (z->s == start)
588             {
589                 current = z;
590             }
591         }
592         //current = home->vec1[0];
593         return;
594     }
595     for (auto z : current->vec1)
596     {
597         if (z->s==str && z->b==true)
598         {
599             current = z;
600             return;
601         }
602     }
603     cout<<"Folder does not exist"<<endl;
604 }
605 void ls()
606 {
607     Drc* dd = current;
608     if (current==root)
609     {
610         cout<<"/"<<endl;
611         //return;
```

```
612     }
613     else
614     {
615         stack<string> st;
616         while(dd!=root)
617         {
618             st.push(dd->s);
619             dd = dd->h;
620         }
621         while(!st.empty())
622         {
623             cout<<"/"<<st.top();
624             st.pop();
625         }
626     }
627     cout<<" $ ls"<<endl;
628     vector<string> folder;
629     vector<string> file;
630     for (auto z : current->vec1)
631     {
632         if (z->b==true)
633         {
634             folder.push_back(z->s);
635         }
636         else
637         {
638             file.push_back(z->s);
639         }
640     }
641     sort(folder.begin(), folder.end());
642     sort(file.begin(), file.end());
643     for (auto z : folder)
644     {
645         cout<<z<<" ";
646     }
647     for (auto z : file)
648     {
649         cout<<z<<" ";
650     }
651     cout<<endl;
652 }
653 void quit()
654 {
655     Drc* dd = current;
656     if (current==root)
657     {
658         cout<<"/"<<endl;
659         //return;
660     }
661     else
662     {
663         stack<string> st;
664         while(dd!=root)
665         {
666             st.push(dd->s);
667             dd = dd->h;
668         }
669         while(!st.empty())
670         {
671             cout<<"/"<<st.top();
672             st.pop();
673         }
674     }
675     cout<<" $ quit"<<endl;
676 }
677
```

```
678 };
679 class Directory_zsh
680 {
681     public:
682     Drc* home;
683     string start;
684     Drc* root;
685     Drc* current;
686     vector<string> vec2;
687     Directory_zsh()
688     {
689         home = new Drc;
690         home->s = "home";
691         string g;
692         cin>>g;
693         start = g;
694         Drc* child = new Drc;
695         child->s = g;
696         child->b = true;
697         (home->vec1).push_back(child);
698         root = new Drc;
699         root->s = "";
700         root->b = true;
701         (root->vec1).push_back(home);
702         child->h = home;
703         home->h = root;
704         current = child;
705     }
706     void pwd()
707     {
708         stack<string> st;
709         string sT = "pwd";
710         vec2.push_back(sT);
711         cout<<(home->vec1)[0]->s<<" ";
712         Drc* dd = current;
713         if (current==root)
714         {
715             cout<<"/"<<endl;
716             //return;
717         }
718         else
719         {
720             while(dd!=root)
721             {
722                 st.push(dd->s);
723                 dd = dd->h;
724             }
725             while(!st.empty())
726             {
727                 cout<<"/"<<st.top();
728                 st.pop();
729             }
730         }
731         cout<<" $ pwd"<<endl;
732         dd = current;
733         if (current==root)
734         {
735             cout<<"/"<<endl;
736             return;
737         }
738         while(dd!=root)
739         {
740             st.push(dd->s);
741             dd = dd->h;
742         }
743         while(!st.empty())
```

```
744     {
745         cout<<"/"<<st.top();
746         st.pop();
747     }
748     cout<<endl;
749 }
750 void mkdir(string str)
751 {
752     cout<<(home->vec1)[0]->s<<" ";
753     string sT = "mkdir ";
754     sT+=str;
755     vec2.push_back(sT);
756     Drc* dd = current;
757     if (current==root)
758     {
759         cout<<"/"<<endl;
760     }
761     else
762     {
763         stack<string> st;
764         while(dd!=root)
765         {
766             st.push(dd->s);
767             dd = dd->h;
768         }
769         while(!st.empty())
770         {
771             cout<<"/"<<st.top();
772             st.pop();
773         }
774     }
775     cout<<" $ mkdir"<<" "<<str<<endl;
776     for (auto z : current->vec1)
777     {
778         if (z->s == str)
779         {
780             if (z->b==true)
781             {
782                 cout<<"Folder Exists"<<endl;
783                 return;
784             }
785             else
786             {
787                 cout<<"File Exists"<<endl;
788                 return;
789             }
790         }
791     }
792     Drc* child = new Drc;
793     child->s = str;
794     child->b = true;
795     child->h = current;
796     (current->vec1).push_back(child);
797 }
798 void touch(string str)
799 {
800     cout<<(home->vec1)[0]->s<<" ";
801     string sT = "touch ";
802     sT+=str;
803     vec2.push_back(sT);
804     Drc* dd = current;
805     if (current==root)
806     {
807         cout<<"/"<<endl;
808         //return;
809     }
```

```
810     else
811     {
812         stack<string> st;
813         while(dd!=root)
814         {
815             st.push(dd->s);
816             dd = dd->h;
817         }
818         while(!st.empty())
819         {
820             cout<<"/"<<st.top();
821             st.pop();
822         }
823     }
824     cout<<" $ touch"<<" "<<str<<endl;
825     for (auto z : current->vec1)
826     {
827         if (z->s == str)
828         {
829             if (z->b==true)
830             {
831                 cout<<"Folder Exists"<<endl;
832                 return;
833             }
834             else
835             {
836                 cout<<"File Exists"<<endl;
837                 return;
838             }
839         }
840     }
841     Drc* child = new Drc;
842     child->s = str;
843     child->b = false;
844     child->h = current;
845     (current->vec1).push_back(child);
846 }
847 void rm(string str)
848 {
849     cout<<(home->vec1)[0]->s<<" ";
850     string sT = "rm ";
851     sT+=str;
852     vec2.push_back(sT);
853     Drc* dd = current;
854     if (current==root)
855     {
856         cout<<"/"<<endl;
857         //return;
858     }
859     else
860     {
861         stack<string> st;
862         while(dd!=root)
863         {
864             st.push(dd->s);
865             dd = dd->h;
866         }
867         while(!st.empty())
868         {
869             cout<<"/"<<st.top();
870             st.pop();
871         }
872     }
873     cout<<" $ rm "<<str<<endl;
874     int ii=0;
875     for (auto z : current->vec1)
```

```
876     {
877         if (z->s == str)
878         {
879             (current->vec1).erase(current->vec1.begin() + ii);
880             return;
881         }
882         ii++;
883     }
884     cout<<"Does not exist"<<endl;
885 }
886 void cd(string str)
887 {
888     cout<<(home->vec1)[0]->s<<" ";
889     string sT = "cd ";
890     sT+=str;
891     vec2.push_back(sT);
892     Drc* dd = current;
893     if (current==root)
894     {
895         cout<<"/"<<endl;
896         //return;
897     }
898     else
899     {
900         stack<string> st;
901         while(dd!=root)
902         {
903             st.push(dd->s);
904             dd = dd->h;
905         }
906         while(!st.empty())
907         {
908             cout<<"/"<<st.top();
909             st.pop();
910         }
911     }
912     cout<<" $ cd " <<str<<endl;
913     if (str=="..")
914     {
915         if (current!=root)
916         {
917             current = current->h;
918         }
919         return;
920     }
921     else if (str=="~")
922     {
923         for (auto z : home->vec1)
924         {
925             if (z->s == start)
926             {
927                 current = z;
928             }
929         }
930         //current = home->vec1[0];
931         return;
932     }
933     for (auto z : current->vec1)
934     {
935         if (z->s==str && z->b==true)
936         {
937             current = z;
938             return;
939         }
940     }
941     cout<<"Folder does not exist"<<endl;
```

```
942     }
943     void ls()
944     {
945         cout<<(home->vec1)[0]->s<<" ";
946         string sT = "ls";
947         vec2.push_back(sT);
948         Drc* dd = current;
949         if (current==root)
950         {
951             cout<<"/"<<endl;
952             //return;
953         }
954         else
955         {
956             stack<string> st;
957             while(dd!=root)
958             {
959                 st.push(dd->s);
960                 dd = dd->h;
961             }
962             while(!st.empty())
963             {
964                 cout<<"/"<<st.top();
965                 st.pop();
966             }
967         }
968         cout<<" $ ls"<<endl;
969         vector<string> folder;
970         vector<string> file;
971         for (auto z : current->vec1)
972         {
973             if (z->b==true)
974             {
975                 folder.push_back(z->s);
976             }
977             else
978             {
979                 file.push_back(z->s);
980             }
981         }
982         sort(folder.begin(), folder.end());
983         sort(file.begin(), file.end());
984         for (auto z : folder)
985         {
986             cout<<z<<" ";
987         }
988         for (auto z : file)
989         {
990             cout<<z<<" ";
991         }
992         cout<<endl;
993     }
994     void quit()
995     {
996         cout<<(home->vec1)[0]->s<<" ";
997         string sT = "quit";
998         vec2.push_back(sT);
999         Drc* dd = current;
1000         if (current==root)
1001         {
1002             cout<<"/"<<endl;
1003             //return;
1004         }
1005         else
1006         {
1007             stack<string> st;
```

```
1008     while(dd!=root)
1009     {
1010         st.push(dd->s);
1011         dd = dd->h;
1012     }
1013     while(!st.empty())
1014     {
1015         cout<<"/"<<st.top();
1016         st.pop();
1017     }
1018     }
1019     cout<<" $ quit"<<endl;
1020 }
1021 void history()
1022 {
1023     cout<<(home->vec1)[0]->s<<" ";
1024     Drc* dd = current;
1025     if (current==root)
1026     {
1027         cout<<"/"<<endl;
1028         //return;
1029     }
1030     else
1031     {
1032         stack<string> st;
1033         while(dd!=root)
1034         {
1035             st.push(dd->s);
1036             dd = dd->h;
1037         }
1038         while(!st.empty())
1039         {
1040             cout<<"/"<<st.top();
1041             st.pop();
1042         }
1043     }
1044     cout<<" $ history"<<endl;
1045     for (int ii=0; ii<vec2.size(); ii++)
1046     {
1047         cout<<ii<<" "<<vec2[ii]<<endl;
1048     }
1049 }
1050
1051 };
1052
1053 int main() {
1054     /* Enter your code here. Read input from STDIN. Print output to STDOUT */
1055     string sr;
1056     string com;
1057     string arg;
1058     cin>>sr;
1059     if (sr=="sh")
1060     {
1061         Directory_sh dS;
1062         while(1)
1063         {
1064             cin>>com;
1065             if (com=="pwd")
1066             {
1067                 dS.pwd();
1068             }
1069             else if (com=="mkdir")
1070             {
1071                 cin>>arg;
1072                 dS.mkdir(arg);
1073             }
1074         }
1075     }
1076 }
```



```
1074     else if (com=="touch")
1075     {
1076         cin>>arg;
1077         dS.touch(arg);
1078     }
1079     else if (com=="rm")
1080     {
1081         cin>>arg;
1082         dS.rm(arg);
1083     }
1084     else if (com=="cd")
1085     {
1086         cin>>arg;
1087         dS.cd(arg);
1088     }
1089     else if (com=="ls")
1090     {
1091         dS.ls();
1092     }
1093     else if (com=="quit")
1094     {
1095         dS.quit();
1096         return 0;
1097     }
1098     else
1099     {
1100         cout<<"$ " <<com<<endl;
1101         cout<<"Command does not exist"<<endl;
1102     }
1103 }
1104 }
1105 else if (sr=="csh")
1106 {
1107     Directory_csh dS;
1108     while(1)
1109     {
1110         cin>>com;
1111         if (com=="pwd")
1112         {
1113             dS.pwd();
1114         }
1115         else if (com=="mkdir")
1116         {
1117             cin>>arg;
1118             dS.mkdir(arg);
1119         }
1120         else if (com=="touch")
1121         {
1122             cin>>arg;
1123             dS.touch(arg);
1124         }
1125         else if (com=="rm")
1126         {
1127             cin>>arg;
1128             dS.rm(arg);
1129         }
1130         else if (com=="cd")
1131         {
1132             cin>>arg;
1133             dS.cd(arg);
1134         }
1135         else if (com=="ls")
1136         {
1137             dS.ls();
1138         }
1139         else if (com=="quit")
```

```
1140     {
1141         dS.quit();
1142         return 0;
1143     }
1144     else
1145     {
1146         cout<<"$ "<<com<<endl;
1147         cout<<"Command does not exist"<<endl;
1148     }
1149 }
1150 }
1151 else if (sr=="bash")
1152 {
1153     Directory_bash dS;
1154     while(1)
1155     {
1156         cin>>com;
1157         if (com=="pwd")
1158         {
1159             dS.pwd();
1160         }
1161         else if (com=="mkdir")
1162         {
1163             cin>>arg;
1164             dS.mkdir(arg);
1165         }
1166         else if (com=="touch")
1167         {
1168             cin>>arg;
1169             dS.touch(arg);
1170         }
1171         else if (com=="rm")
1172         {
1173             cin>>arg;
1174             dS.rm(arg);
1175         }
1176         else if (com=="cd")
1177         {
1178             cin>>arg;
1179             dS.cd(arg);
1180         }
1181         else if (com=="ls")
1182         {
1183             dS.ls();
1184         }
1185         else if (com=="quit")
1186         {
1187             dS.quit();
1188             return 0;
1189         }
1190         else
1191         {
1192             Drc* dd = dS.current;
1193             if (dS.current==dS.root)
1194             {
1195                 cout<<"/"<<endl;
1196                 return 0;
1197             }
1198             stack<string> st;
1199             while(dd!=dS.root)
1200             {
1201                 st.push(dd->s);
1202                 dd = dd->h;
1203             }
1204             while(!st.empty())
1205             {
```

```
1206         cout<<"/"<<st.top();
1207         st.pop();
1208     }
1209     cout<<" $ "<<com<<endl;
1210     cout<<"Command does not exist"<<endl;
1211 }
1212 }
1213
1214 }
1215 else if (sr=="zsh")
1216 {
1217     Directory_zsh dS;
1218     while(1)
1219     {
1220         cin>>com;
1221         if (com=="pwd")
1222         {
1223             dS.pwd();
1224         }
1225         else if (com=="mkdir")
1226         {
1227             cin>>arg;
1228             dS.mkdir(arg);
1229         }
1230         else if (com=="touch")
1231         {
1232             cin>>arg;
1233             dS.touch(arg);
1234         }
1235         else if (com=="rm")
1236         {
1237             cin>>arg;
1238             dS.rm(arg);
1239         }
1240         else if (com=="cd")
1241         {
1242             cin>>arg;
1243             dS.cd(arg);
1244         }
1245         else if (com=="ls")
1246         {
1247             dS.ls();
1248         }
1249         else if (com=="quit")
1250         {
1251             dS.quit();
1252             return 0;
1253         }
1254         else if (com=="history")
1255         {
1256             dS.history();
1257         }
1258         else
1259         {
1260             Drc* dd = dS.current;
1261             if (dS.current==dS.root)
1262             {
1263                 cout<<"/"<<endl;
1264                 return 0;
1265             }
1266             stack<string> st;
1267             while(dd!=dS.root)
1268             {
1269                 st.push(dd->s);
1270                 dd = dd->h;
1271             }
```

```
1272         while(!st.empty())
1273     {
1274         cout<<"/"<<st.top();
1275         st.pop();
1276     }
1277     dS.vec2.push_back(com);
1278     cout<<" $ "<<com<<endl;
1279     cout<<"Command does not exist"<<endl;
1280     }
1281 }
1282
1283 }
1284 return 0;
1285 }
1286 }
```

Line: 1 Col: 1

 [Upload Code as File](#) ☐ Test against custom input

Run Code

Submit Code

[Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) |