Terminal Simulation

locked

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
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

```
csh 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
```

Submissions: 83
Max Score: 100
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

C++20

```
#include <cmath>
#include <cstdio>
#include <stack>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
struct Drc
{
    string s;
    Drc* h;
    bool b;
    vector<Drc*> vec1;
};
class Directory_sh
{
    public:
```

```cpp
    Drc* home;
    string start;
    Drc* root;
    Drc* current;
    Directory_sh()
    {
        home = new Drc;
        home->s = "home";
        string g;
        cin>>g;
        start = g;
        Drc* child = new Drc;
        child->s = g;
        child->b = true;
        (home->vec1).push_back(child);
        root = new Drc;
        root->s = "";
        root->b = true;
        (root->vec1).push_back(home);
        child->h = home;
        home->h = root;
        current = child;
    }
    void pwd()
    {
        cout<<"$ pwd"<<endl;
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            return;
        }
        stack<string> st;
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        cout<<endl;
    }
    void mkdir(string str)
    {
        cout<<"$ mkdir"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = true;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void touch(string str)
    {
        cout<<"$ touch"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = false;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void rm(string str)
    {
        cout<<"$ rm "<<str<<endl;
        int ii=0;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                (current->vec1).erase(current->vec1.begin() + ii);
                return;
            }
            ii++;
        }
        cout<<"Does not exist"<<endl;
    }
    void cd(string str)
    {
        cout<<"$ cd "<<str<<endl;
        if (str=="..")
        {
            if (current!=root)
            {
                current = current->h;
            }
            return;
        }
        for (auto z : current->vec1)
        {
            if (z->s==str && z->b==true)
            {
                current = z;
                return;
            }
        }
        cout<<"Folder does not exist"<<endl;
    }
    void ls()
    {
        cout<<"$ ls"<<endl;
        vector<string> folder;
        vector<string> file;
        for (auto z : current->vec1)
        {
            if (z->b==true)
            {
                folder.push_back(z->s);
            }
            else
            {
                file.push_back(z->s);
            }
        }
        sort(folder.begin(), folder.end());
        sort(file.begin(), file.end());
        for (auto z : folder)
        {
            cout<<z<<" ";
        }
        for (auto z : file)
        {
            cout<<z<<" ";
        }
        cout<<endl;
    }
    void quit()
    {
        cout<<"$ quit"<<endl;
    }
};
class Directory_csh
{
    public:
    Drc* home;
    string start;
    Drc* root;
    Drc* current;
    Directory_csh()
    {
        home = new Drc;
        home->s = "home";
        string g;
        cin>>g;
        start = g;
        Drc* child = new Drc;
        child->s = g;
        child->b = true;
        (home->vec1).push_back(child);
        root = new Drc;
        root->s = "";
        root->b = true;
        (root->vec1).push_back(home);
        child->h = home;
        home->h = root;
        current = child;
    }
    void pwd()
    {
        cout<<"$ pwd"<<endl;
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            return;
        }
        stack<string> st;
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        cout<<endl;
    }
    void mkdir(string str)
    {
        cout<<"$ mkdir"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = true;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void touch(string str)
    {
        cout<<"$ touch"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = false;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void rm(string str)
    {
        cout<<"$ rm "<<str<<endl;
        int ii=0;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                (current->vec1).erase(current->vec1.begin() + ii);
                return;
            }
            ii++;
        }
        cout<<"Does not exist"<<endl;
    }
    void cd(string str)
    {
        cout<<"$ cd "<<str<<endl;
        if (str=="..")
        {
            if (current!=root)
            {
                current = current->h;
            }
            return;
        }
        else if (str=="~")
        {
            for (auto z : home->vec1)
            {
                if (z->s == start)
                {
                    current = z;
                }
            }
            //current = home->vec1[0];
            return;
        }
        for (auto z : current->vec1)
        {
            if (z->s==str && z->b==true)
            {
                current = z;
                return;
            }
        }
        cout<<"Folder does not exist"<<endl;
    }
    void ls()
    {
        cout<<"$ ls"<<endl;
        vector<string> folder;
        vector<string> file;
        for (auto z : current->vec1)
        {
            if (z->b==true)
            {
                folder.push_back(z->s);
            }
            else
            {
                file.push_back(z->s);
            }
        }
        sort(folder.begin(), folder.end());
        sort(file.begin(), file.end());
        for (auto z : folder)
        {
            cout<<z<<" ";
        }
        for (auto z : file)
        {
            cout<<z<<" ";
        }
        cout<<endl;
    }
    void quit()
    {
        cout<<"$ quit"<<endl;
    }
};
class Directory_bash
{
    public:
    Drc* home;
    string start;
    Drc* root;
    Drc* current;
    Directory_bash()
    {
        home = new Drc;
        home->s = "home";
        string g;
        cin>>g;
        start = g;
        Drc* child = new Drc;
        child->s = g;
        child->b = true;
        (home->vec1).push_back(child);
        root = new Drc;
        root->s = "";
        root->b = true;
        (root->vec1).push_back(home);
        child->h = home;
        home->h = root;
        current = child;
    }
    void pwd()
    {
        Drc* dd = current;
        stack<string> st;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        }
        cout<<" $ pwd"<<endl;
        dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            return;
        }
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        cout<<endl;
    }
    void mkdir(string str)
    {
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ mkdir"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = true;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void touch(string str)
    {
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        {
        stack<string> st;
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        }
        cout<<" $ touch"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = false;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void rm(string str)
    {
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ rm "<<str<<endl;
        int ii=0;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                (current->vec1).erase(current->vec1.begin() + ii);
                return;
            }
            ii++;
        }
        cout<<"Does not exist"<<endl;
    }
    void cd(string str)
    {
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
        stack<string> st;
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        }
        cout<<" $ cd "<<str<<endl;
        if (str=="..")
        {
            if (current!=root)
            {
                current = current->h;
            }
            return;
        }
        else if (str=="~")
        {
            for (auto z : home->vec1)
            {
                if (z->s == start)
                {
                    current = z;
                }
            }
            //current = home->vec1[0];
            return;
        }
        for (auto z : current->vec1)
        {
            if (z->s==str && z->b==true)
            {
                current = z;
                return;
            }
        }
        cout<<"Folder does not exist"<<endl;
    }
    void ls()
    {
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ ls"<<endl;
        vector<string> folder;
        vector<string> file;
        for (auto z : current->vec1)
        {
            if (z->b==true)
            {
                folder.push_back(z->s);
            }
            else
            {
                file.push_back(z->s);
            }
        }
        sort(folder.begin(), folder.end());
        sort(file.begin(), file.end());
        for (auto z : folder)
        {
            cout<<z<<" ";
        }
        for (auto z : file)
        {
            cout<<z<<" ";
        }
        cout<<endl;
    }
    void quit()
    {
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ quit"<<endl;
    }

```

```
};
class Directory_zsh
{
    public:
    Drc* home;
    string start;
    Drc* root;
    Drc* current;
    vector<string> vec2;
    Directory_zsh()
    {
        home = new Drc;
        home->s = "home";
        string g;
        cin>>g;
        start = g;
        Drc* child = new Drc;
        child->s = g;
        child->b = true;
        (home->vec1).push_back(child);
        root = new Drc;
        root->s = "";
        root->b = true;
        (root->vec1).push_back(home);
        child->h = home;
        home->h = root;
        current = child;
    }
    void pwd()
    {
        stack<string> st;
        string sT = "pwd";
        vec2.push_back(sT);
        cout<<(home->vec1)[0]->s<<" ";
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ pwd"<<endl;
        dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            return;
        }
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        cout<<endl;
    }
    void mkdir(string str)
    {
        cout<<(home->vec1)[0]->s<<" ";
        string sT = "mkdir ";
        sT+=str;
        vec2.push_back(sT);
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ mkdir"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = true;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void touch(string str)
    {
        cout<<(home->vec1)[0]->s<<" ";
        string sT = "touch ";
        sT+=str;
        vec2.push_back(sT);
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ touch"<<" "<<str<<endl;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                if (z->b==true)
                {
                    cout<<"Folder Exists"<<endl;
                    return;
                }
                else
                {
                    cout<<"File Exists"<<endl;
                    return;
                }
            }
        }
        Drc* child = new Drc;
        child->s = str;
        child->b = false;
        child->h = current;
        (current->vec1).push_back(child);
    }
    void rm(string str)
    {
        cout<<(home->vec1)[0]->s<<" ";
        string sT = "rm ";
        sT+=str;
        vec2.push_back(sT);
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ rm "<<str<<endl;
        int ii=0;
        for (auto z : current->vec1)
        {
            if (z->s == str)
            {
                (current->vec1).erase(current->vec1.begin() + ii);
                return;
            }
            ii++;
        }
        cout<<"Does not exist"<<endl;
    }
    void cd(string str)
    {
        cout<<(home->vec1)[0]->s<<" ";
        string sT = "cd ";
        sT+=str;
        vec2.push_back(sT);
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ cd "<<str<<endl;
        if (str=="..")
        {
            if (current!=root)
            {
                current = current->h;
            }
            return;
        }
        else if (str=="~")
        {
            for (auto z : home->vec1)
            {
                if (z->s == start)
                {
                    current = z;
                }
            }
            //current = home->vec1[0];
            return;
        }
        for (auto z : current->vec1)
        {
            if (z->s==str && z->b==true)
            {
                current = z;
                return;
            }
        }
        cout<<"Folder does not exist"<<endl;
    }
    void ls()
    {
        cout<<(home->vec1)[0]->s<<" ";
        string sT = "ls";
        vec2.push_back(sT);
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
            stack<string> st;
            while(dd!=root)
            {
                st.push(dd->s);
                dd = dd->h;
            }
            while(!st.empty())
            {
                cout<<"/"<<st.top();
                st.pop();
            }
        }
        cout<<" $ ls"<<endl;
        vector<string> folder;
        vector<string> file;
        for (auto z : current->vec1)
        {
            if (z->b==true)
            {
                folder.push_back(z->s);
            }
            else
            {
                file.push_back(z->s);
            }
        }
        sort(folder.begin(), folder.end());
        sort(file.begin(), file.end());
        for (auto z : folder)
        {
            cout<<z<<" ";
        }
        for (auto z : file)
        {
            cout<<z<<" ";
        }
        cout<<endl;
    }
    void quit()
    {
        cout<<(home->vec1)[0]->s<<" ";
        string sT = "quit";
        vec2.push_back(sT);
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
        stack<string> st;
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        }
        cout<<" $ quit"<<endl;
    }
    void history()
    {
        cout<<(home->vec1)[0]->s<<" ";
        Drc* dd = current;
        if (current==root)
        {
            cout<<"/"<<endl;
            //return;
        }
        else
        {
        stack<string> st;
        while(dd!=root)
        {
            st.push(dd->s);
            dd = dd->h;
        }
        while(!st.empty())
        {
            cout<<"/"<<st.top();
            st.pop();
        }
        }
        cout<<" $ history"<<endl;
        for (int ii=0; ii<vec2.size(); ii++)
        {
            cout<<ii<<" "<<vec2[ii]<<endl;
        }
    }

};

int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    string sr;
    string com;
    string arg;
    cin>>sr;
    if (sr=="sh")
    {
      Directory_sh dS;
      while(1)
      {
          cin>>com;
          if (com=="pwd")
          {
              dS.pwd();
          }
          else if (com=="mkdir")
          {
              cin>>arg;
              dS.mkdir(arg);
          }
          else if (com=="touch")
          {
              cin>>arg;
              dS.touch(arg);
          }
          else if (com=="rm")
          {
              cin>>arg;
              dS.rm(arg);
          }
          else if (com=="cd")
          {
              cin>>arg;
              dS.cd(arg);
          }
          else if (com=="ls")
          {
              dS.ls();
          }
          else if (com=="quit")
          {
              dS.quit();
              return 0;
          }
          else
          {
              cout<<"$ "<<com<<endl;
              cout<<"Command does not exist"<<endl;
          }
      }
    }
    else if (sr=="csh")
    {
      Directory_csh dS;
      while(1)
      {
          cin>>com;
          if (com=="pwd")
          {
              dS.pwd();
          }
          else if (com=="mkdir")
          {
              cin>>arg;
              dS.mkdir(arg);
          }
          else if (com=="touch")
          {
              cin>>arg;
              dS.touch(arg);
          }
          else if (com=="rm")
          {
              cin>>arg;
              dS.rm(arg);
          }
          else if (com=="cd")
          {
              cin>>arg;
              dS.cd(arg);
          }
          else if (com=="ls")
          {
              dS.ls();
          }
          else if (com=="quit")
```

```
            {
                dS.quit();
                return 0;
            }
            else
            {
                cout<<"$ "<<com<<endl;
                cout<<"Command does not exist"<<endl;
            }
        }
    }
    else if (sr=="bash")
    {
      Directory_bash dS;
      while(1)
      {
          cin>>com;
          if (com=="pwd")
          {
              dS.pwd();
          }
          else if (com=="mkdir")
          {
              cin>>arg;
              dS.mkdir(arg);
          }
          else if (com=="touch")
          {
              cin>>arg;
              dS.touch(arg);
          }
          else if (com=="rm")
          {
              cin>>arg;
              dS.rm(arg);
          }
          else if (com=="cd")
          {
              cin>>arg;
              dS.cd(arg);
          }
          else if (com=="ls")
          {
              dS.ls();
          }
          else if (com=="quit")
          {
              dS.quit();
              return 0;
          }
          else
          {
              Drc* dd = dS.current;
              if (dS.current==dS.root)
              {
                cout<<"/"<<endl;
                return 0;
              }
              stack<string> st;
              while(dd!=dS.root)
              {
                st.push(dd->s);
                dd = dd->h;
              }
              while(!st.empty())
              {
                  cout<<"/"<<st.top();
                  st.pop();
                }
                cout<<" $ "<<com<<endl;
                cout<<"Command does not exist"<<endl;
            }
        }

      }
      else if (sr=="zsh")
      {
        Directory_zsh dS;
        while(1)
        {
            cin>>com;
            if (com=="pwd")
            {
                dS.pwd();
            }
            else if (com=="mkdir")
            {
                cin>>arg;
                dS.mkdir(arg);
            }
            else if (com=="touch")
            {
                cin>>arg;
                dS.touch(arg);
            }
            else if (com=="rm")
            {
                cin>>arg;
                dS.rm(arg);
            }
            else if (com=="cd")
            {
                cin>>arg;
                dS.cd(arg);
            }
            else if (com=="ls")
            {
                dS.ls();
            }
            else if (com=="quit")
            {
                dS.quit();
                return 0;
            }
            else if (com=="history")
            {
                dS.history();
            }
            else
            {
                Drc* dd = dS.current;
                if (dS.current==dS.root)
                {
                  cout<<"/"<<endl;
                  return 0;
                }
                stack<string> st;
                while(dd!=dS.root)
                {
                  st.push(dd->s);
                  dd = dd->h;
                }
                while(!st.empty())
                {
                  cout<<"/"<<st.top();
                  st.pop();
                }
                dS.vec2.push_back(com);
                cout<<" $ "<<com<<endl;
                cout<<"Command does not exist"<<endl;
            }
        }

    }
    return 0;
}

```

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 |