Write a regular expression that matches on a string with two vowel followed by the letter “s” or “r”.

The following regular expression matches on what kind of string: '^[123456790]\{4\}$'

Why must a script start with the #! symbol?

What command is used to read data from STDIN?

When running a command, what variable contains the first argument of that command?

How does one test if two numbers are equal?

What services does an operating system provide?

What command prints out the current directory?
If statements have a syntax similar to Java, but without brackets.

```plaintext
if _condition_
then
   _code_
elif _condition_
then
   _code_
else
   _code_
fi
```
The following example program can be used to add or subtract two numbers.

```bash
#!/bin/sh
if test $1 = add
then
    result=`expr $2 + $3`
elif test $1 = sub
then
    result=`expr $2 - $3`
else
    result=0
fi
echo "The result is $result \n"
```
A case statement is similar to a Java switch statement.

case _condition_ in
    _condition1_) _action1_;;
    _condition2_) _action2_;;
    _condition3_ | _condition4_) _action3_;;
    *) _else_action_;;
esac
The following example program is a remodeling of the if example, but with a case statement.

```
#!/bin/sh
case $1 in
  add | addition)
    result=`expr $2 + $3`;;
  sub | substraction)
    result=`expr $2 - $3`;;
  *)
    result=0;;
esac
echo "The result is $result \n"
```
The for loop is similar to a Java iterator. It allows you to iterate (loop) over a list of strings.

```python
for _var_ in _list_
do
    _action_
done
```
The following script executes the file command for each file in the specified path.

```bash
#!/usr/bin/sh
for i in `ls $1`
do
   file $i;
done
```
The last control statement we will need is the while statement. Again, it is very similar to its Java equivalent.

```c
while _condition_
  _action_
[continue]
[break]
end
```
The following script will pad a file with zeros.

```bash
#!/usr/bin/sh

i=`wc -c < $1`
while test $i -lt $2
do
echo -n "0" >> $1;
i=`wc -c < $1`
done
```
Some commands, such as date, have output that require an extra bit of parsing to use.

Sun Aug 13 11:42:38 EDT 2006

- You can use the set command to capture and parse the output.

    set 'date'

- The output will be stored in $n ($1, $2, $3, etc).

- Note that using set will erase any data you might already have in $n.
The following script executes the date command and outputs the parsed result.

```bash
#!/usr/bin/sh
set 'date'
echo "Time: $4 $5"
echo "Day: $1"
echo "Date: $3 $2 $6"
```

The output would be as follows:

```
Time: 12:45:54 EDT
Day: Sun
Date: 13 Aug 2006
```
Uses of script files

- **Backup scripts**
  - archiving important files and saving them in a safe place.

- **Startup scripts**
  - application which require a complicated environment to run.

- **Scheduled scripts**
  - regularly rotating and archiving logs.

- **Maintenance scripts**
  - a script that creates a user or changes a password on multiple system, all in one command.
When you login or start a shell, specific scripts are executed to configure your environment.

The specific script depends on which shell you are using.

- With Bash, `~/.bash_profile` is executed on login.
- With csh (and its derivative, like tcsh), the `/.cshrc` file is executed.

While customizing your account, you might want to set some variable, such as the PATH and the CLASSPATH.
The PATH is a set of directory a shell searches for executables.
- On Unix, it’s a colon (:) separated list.
- On Windows, it’s a semi-colon (;) separated list.
- You can use the which command to figure out what file will be executed.

The CLASSPATH is the set of directory the JVM searches when loading classes.
Changing an environment variable

- You can output an environment variable using `echo`.
- Before something goes wrong, you might want to backup the old value.
  ```bash
  TEMP_VAR_NAME=${VAR_TO_CHANGE}
  ```
- You can then set the new value.
  ```bash
  VAR_TO_CHANGE=SomethingElse
  ```
- This works in bash. In tcsh, you need to use `setenv`.
- You can use any existing variables.
  ```bash
  VAR_TO_CHANGE=${VAR_TO_CHANGE}:SomethingElse
  ```
- If something goes wrong, you can revert to the old value.
  ```bash
  VAR_TO_CHANGE=${TEMP_VAR_NAME}
  ```
Other startup things . . .

- You can customize your command prompt by changing an environment variable.
  - Check your shell’s documentation for information on how to do this.

- You can use the alias command to set up shortcuts.
  alias ll='/usr/bin/ls -l'

- You can set your default editor (CVS uses this).
  EDITOR=vi

- Some applications might require you to set up an environment variable.
  PVM_ROOT=/usr/local
Public key cryptography is a form of cryptography which generally allows users to communicate securely without having prior access to a shared secret key.

- the private key is kept secret
- the public key may be widely distributed