1. Wildcards are special characters which are used to replace or represent one or more characters.  
a) True  
b) False

Answer: a  
Explanation: A wildcard is a special character which can be used as a substitute for any of a class of characters, which increases the flexibility and efficiency of searching and replacing. For example, to remove all the files with a filename starting with ‘chap’ prefix, we can use the command rm chap\*. Here \* is used as a wildcard for matching all filenames starting with ‘chap’.

2. Which of the following is not a wild-card?  
a) \*  
b) ?  
c) $  
d) %

Answer: c  
Explanation: The ‘$’ sign represents the shell prompt while all the other characters belong to a category of shell wildcards.

‘\*’ // matches any number of characters

‘?’ // matches a single character

3. What does the following command do?

$ echo \*

a) error  
b) undefined behavior  
c) displays “\*”  
d) lists all filenames in the current directory

Answer: d  
Explanation: When we use echo command with only \* as the argument we simply see a list of file. All the filenames in the current directory are displayed on the terminal. Since we know that \* is a wildcard that can match any number of character. Here it is used as solitary to match all filenames.

4. Which command would be most suitable to remove the following files?

dirx diry dirz dirzw

a) rm dir?  
b) rm dirx diry dirz dirzw  
c) rm \*  
d) rm dir\*

Answer: a  
Explanation: Since we know that ? can be used to match a single character. In the above scenario, all the filenames are same except that the last character in all filenames is different. So we can use the ? meta-character.

5. Which of the following files will not be deleted using “rm chap??” ?  
a) chap01  
b) chap02  
c) chaptd  
d) chactd

Answer: d  
Explanation: Since ? is used to match a single character, ?? can match two characters. So the above command will remove all files with a filename starting with a prefix ‘chap’, followed by any two characters.

6. Which of the following command will list all the hidden filenames in our directory having at least three characters after the dot (.)?  
a) ls  
b) ls -a  
c) ls .???\*  
d) ls \*

Answer: c  
Explanation: The \* doesn’t match all files beginning with a (.) dot. So if we need to lists all the hidden filenames in our directory having at least three characters after the dot (.) we can use the following command,

$ ls .???\*

.bash\_pro .chap01 .netspak .profile //list of hidden files

7. \* and ? cannot match \_\_\_\_  
a) /  
b) $  
c) .  
d) / and .

Answer: d  
Explanation: There are two things which \* and ? cannot match. First is, they cannot match filenames starting with a dot (.). Second is, they cannot match / in the pathname. For example, we cannot use cd /usr?local to switch to /usr/local. It will generate an error.

8. rm chap0[1234] will delete all of the following files.

chap01 chap02 chap03 chap04

a) True  
b) False

Answer: a  
Explanation: We can frame restrictive patterns with the character class. The character class comprises a set of characters enclosed by the rectangular brackets, [ and ], but it matches a single character in the class. For example, the pattern [abcd] is a character class that can match a single character – an a, b, c or d. Similarly, we can combine the character class with any string or any other wildcard expression. Hence, the command rm chap0[1234] will delete chap01 chap02 chap03 chap04.

9. Which of the following files will not be listed using the following command?

ls chap0[1-4]

a) chap02  
b) chap05  
c) chap01  
d) chap04

Answer: b  
Explanation: Range specification is also possible inside the class using a hyphen (-). The two characters on either side of it form the range of characters to be matched. So the command ls chap0[1-4] can match chap01, chap02, chap03, chap04. But chap05 cannot be matched because the range inside the character class is from 1 to 4.

10. Which of the following symbol is used for negating the character class?  
a) .  
b) \*  
c) !  
d) %

Answer: c  
Explanation: We can use the ! as the first symbol for negating the character class. For example,

\*.[!tx] //matches all filenames with single character extension but not .t or .x

[!a-zA-Z]\* // matches all filenames that don’t begin with a alphabetic character.

11. Providing a backslash (\) before the wild card to remove its special meaning is called \_\_\_\_\_  
a) escaping  
b) quoting  
c) listing  
d) pattern matching

Answer: a  
Explanation: We know that shell uses some special characters to match filenames or perform other search and replace operations. But if the filename itself contains those special characters, then it could be a great nuisance. For dealing with such files we use escaping and quoting.  
Escaping means providing a backslash (\) before the wild card so that its special meaning could be removed. For example, if we want to remove a file named chap\*, then using the command rm chap\* will delete all the file beginning with a prefix ’chap’. In this situation, we can use the following command,

$ rm chap\\*

12. Enclosing the wild card or the entire pattern within quotes is called \_\_\_  
a) escaping  
b) quoting  
c) listing  
d) pattern matching

Answer: b  
Explanation: There is another way to turn off the meaning of the metacharacter. When a command argument is enclosed in quotes, the meaning of all special characters is turned off. This method is called quoting. For example,

$ echo ‘\’ //displays a \

13. To remove the file named my document.txt, which one of the following commands will be used?  
a) rm my\ document.txt  
b) rm my document.txt  
c) rm \*  
d) rm my\_document.txt

Answer: a  
Explanation: Apart from metacharacters, there are other characters that are special like the space character. The shell uses it to delimit command line arguments. So to remove the file my document.txt we can use escaping. The backslash will make the shell to ignore the space. Hence file will be removed easily.

14. We can escape the \ itself using escaping.  
a) True  
b) False

Answer: a  
Explanation: Sometimes we may need to interpret the \ itself literally. To do so we need another \ before it. For example,

$ echo \\ // displays \

15. For escaping the newline character we can use \_\_\_\_  
a) /  
b) \  
c) ?  
d) \n

Answer: b  
Explanation: The newline character is also special, it marks the end of the command line. Some command lines that uses several arguments can be long enough to overflow to the next line. To split the wrapped line into two lines, make sure you input a \ before pressing [Enter].

16. The output of the following command is \_\_\_\_\_\_\_\_\_\_\_\_\_\_

$ echo ‘The special character $ echo hello and | ls chap\*’

a) undefined output  
b) erroneous  
c) hello  
d) The special character $ echo hello and | ls chap\*

Answer: d  
Explanation: Quoting is another way to turn off the meaning of a meta-character. When a command argument is enclosed in quotes, the meanings of all enclosed special characters are turned off.

17. Double quotes are more permissive than single quotes and allow the evaluation of the $ and ` itself.  
a) True  
b) False

Answer: a  
Explanation: When a command argument is enclosed in single quotes, the meaning of all enclosed special characters is turned off. But when we enclose the same in double quotes, we cannot protect the $ and ` (backquote). For example,

$ echo ‘$SHELL’ // displays $SHELL

$ echo “$SHELL” // evaluates $SHELL and then display

18. Which of the following command will remove the file named \* ?  
a) rm \*  
b) rm ‘\*’  
c) rm \\*  
d) rm ‘\*’ and rm \\*

Answer: d  
Explanation: To suppress the nature of wildcard \* we can use either escaping or quoting. The \ symbol will suppress the feature of wildcard and will remove the file named \*. Similarly using quoting we can turn off the meaning of the meta character.