1. The differences between a text file and a binary file in the context of Git is that binary files have meaningless diffs between versions where Git will show differences between text files and can automatically merge changes from different branches when possible. Also, text files are stored as a series of lines where Git can track changes at the line level whereas binary files are stored as blobs where changes cannot be tracked.
2. A repository in the context of version control is a storage location for the working tree, store, and index of a project. The working tree consists of the project itself (directory with files and subdirectories), the store consists of the history of the project (hidden directory), and the index is the virtual snapshot of the project.
3. A local repository is one that is stored on your local machine where you make changes, commits, and update branches locally. A remote repository is one that is hosted on a remote server. This serves as a central repository that many people can clone and work on from different locations.
4. To clone a repository in Git, you first open your terminal or command prompt and go to the directory where you want to clone the repository. Obtain the URL of the remote repository and type 'git clone <The URL of the repository>' in the command line.
5. The purpose of a .gitignore file is to specify which files and directories from the working tree should be ignored by Git.
6. A commit in Git is a snapshot of the changes in a repository at a certain point in time. A commit can be viewed as both a complete snapshot of the project at that point in time (a revision), and a delta of the changes made (a patch); that is, the diff between revisions.
7. To see the status of your repository in Git, on the command line, type 'git status', and it will tell you the current branch you are on and whether or not your branch is up to date with the remote branch.

8. A pull request is a request to merge changes from one branch into another.
9. Git merge is a command used to combine changes from one branch into another. You would use git merge when you have created a new feature or fixed a bug in a separate branch and you want to integrate it into the main branch.
10. To resolve conflicts when merging branches in Git, you first need to identify the conflict. Then, edit the files to resolve the conflicts, add the resolved file, and finally commit the changes. To avoid conflicts arising, make sure to communicate with collaborators, make frequent pulls from the remote repository, make smaller and focused changes, and use feature branches to minimize conflict risk.
11. The difference between git pull and git fetch is that git fetch just fetches updates from the remote repository without merging them into your current branch. Git pull, on the other hand, is a combination of both git pull and git merge, so it fetches updates from the remote branch and then merges them into your current branch.
12. There are several ways to revert changes in Git depending on what you want to revert.
    a. To discard changes in a file and revert it to the last committed stage, type ‘git checkout – filename'
    b. To discard changes to all files in a working tree, type ‘git checkout --’
    c. To unstage a file from the staging area, type ‘git reset HEAD filename’
    d. To unstage all files from the staging area, type ‘git reset HEAD .
    e. To create a new commit that undoes the changes made by a certain commit, type ‘git revert commit-hash'
13. To view the commit history of a Git repository, type ‘git log’, which will show a list of commits, their hash, commit message, authors, and timestamps. To visualize the commit history as a graph, type ‘git log --graph'.

14. A Git branch is a movable pointer to a commit. To create a branch, type ‘git branch branchname2’. To merge branches, switch to the branch you want to merge the changes to, ‘git checkout branchname1’, and use the git merge command to merge the changes from the feature branch to the target branch, ‘git merge branchname2’.
15. A normal repository includes a working tree, where a bare repository does not. In a normal repository, files can be directly edited, stages, and committed, whereas in a bare repository, files can not be directly edited.
16. Using the given diagram:
    a. To create another commit in the feature branch, first pull from the remote repository, ‘git pull’, then make a commit on the feature branch, ‘git commit’, and finally push the commit to the remote ‘git push origin feature’.
    b. To create a new branch pointing to the latest snapshot on the master branch, first switch to the main branch ‘git checkout master’, then make sure your local repository is up to date, ‘git pull origin master’, and finally create a branch, ‘git branch snapshot’.
    c. To pull the latest changes from the remote repository to your local repository, you type ‘git pull’.
    d. To push your code to the remote repository, first add the files to the staging area ‘git add .’, then commit your changes ‘git commit’, and finally push your changes to the remote repository, ‘git push origin master’.
    e. The master branch does not contain all the source code, as there is a feature branch with separate code that has not yet been merged with the master branch.

f. To merge the feature branch into the master branch, first switch the head to the master branch, ‘git checkout master’. Then, merge the branches together with ‘git merge feature’.

17. False, EVERYTHING in Ruby is an object.

18. False, ‘a’ is “smaller” than ‘b’ since a comes before b so it would print –1, not 1.

19. True, the least significant bit of the reference is used to indicate whether the value is an immediate value or a reference to an object.

20. True, falsy values include only false and nil.

21. The output of the code:

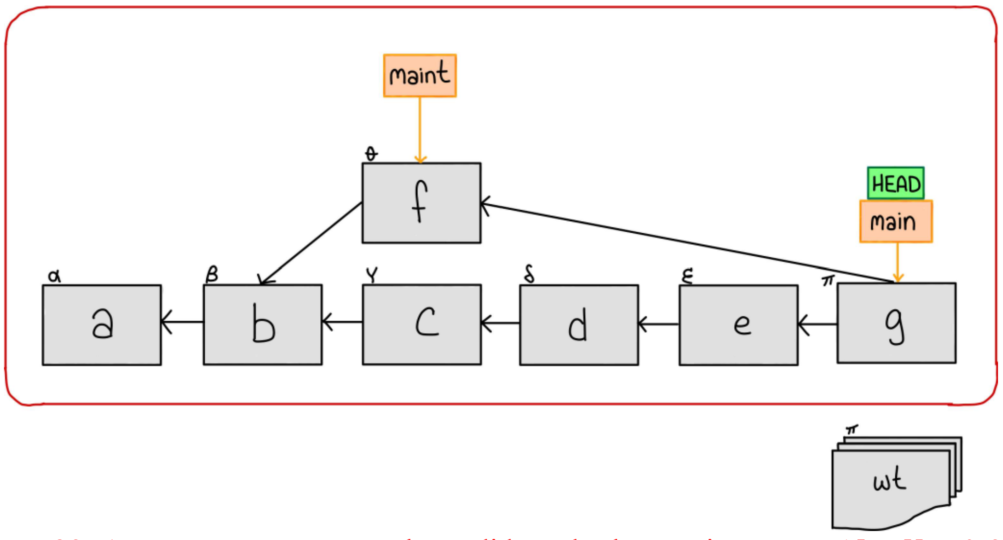
a. The output of the first part of the code would be 3 followed by a runtime error due to trying to modify a frozen array.

b. The output of the second part of the code would be ABCDE

22. Using the diagram given:

a. To merge the maint branch into the main branch, first move the head to main branch ‘git checkout main’. Then, merge the two branches ‘git merge maint’.

b.



23. A regex pattern to match a valid method name is regex = ^[a-z][a-z0-9_]*$

a. The ^ asserts the start of the string

b. [a-z] matches any sing lowercase letter

c. [a-z0-9_]* matches zero or more lowercase letters, numbers, or underscores

d. $ asserts the end of the string

24. The output of the Ruby code snippet will be "y/x=-3"

    a. Remember that Ruby rounds to negative infinity

25. The output of the Ruby code snippet will be [20,30,5,6,7,8]

    a. Remember that .shift removes an element from the front of the array

26. The output of the Ruby code snippet will be [1,2,1,9,3]

27. The output of the Ruby code snippet will be true

28. Here are some email addresses that will match the regular expression given:

    a. Sarah_smith.34@osu.edu

    b. garywhite.21@osu.edu

    c. A_j.2@osu.edu

29. Here is the corresponding language for each of the following:

    a. {"cat", "dog", "fish"}

    b. {"hello", "Hello"}

    c. {"Ruby", "Rails"}

    d. {"Gray", "Grey", "gray", "grey"}

30. Here is a description of the language for each:

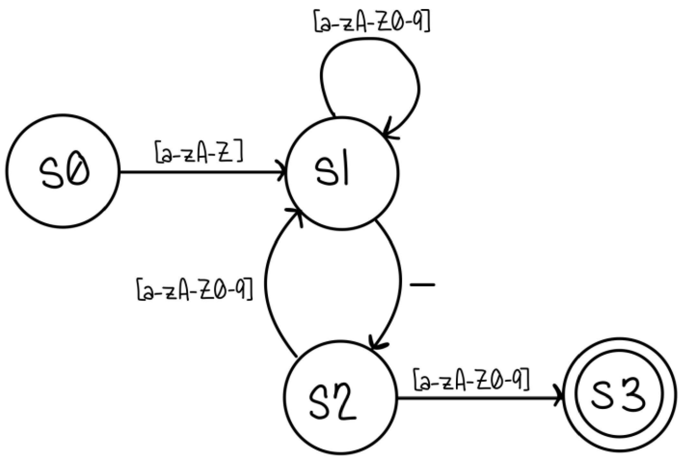
    a. This will be a short set of strings that consists of {"Gray", "Grey"}. The strings will start with 'Gr', followed by either 'a' or 'e' and ends with 'y'

    b. This will be a set of strings that start with a 0, followed by either a lowercase or uppercase 'x', followed be a single hexadecimal digit. (EX: 0x3)

c. This will be a set of strings that start with either a uppercase or lowercase ‘q’, followed by a bracket ‘]’, and ends with any character except for ‘u’. (EX: q]A)

31. Language contains only letters, numbers, and _, starts with a letter, does not contain 2 consecutives _’s and does not end with a _.

a. The corresponding RE is ^[a-zA-Z][a-zA-Z0-9]*(?:_[a-zA-Z0-9]+)*$



b. The corresponding FSA:

32. To convert Man to base64 (TWFu):

a. First, find the ASCII values for each character

i. M has ASCII value of 77

ii. a has ASCII value of 97

iii. n has ASCII value of 110

b. Convert each ASCII value to binary

i. M(77) to binary is 01001101

ii. a(97) to binary is 01100001

iii. n(110) to binary is 01101110

c. Group the entire binary into 6-bit chunks

i. 010011 010110 000101 101110

d. Convert each 6-bit chunk to decimal\

    i. 010011 = 19
    ii. 010110 = 22
    iii. 000101 = 5
    iv. 101110 = 46

e. Finally, map decimal values to Base64 characters
    i. 19 = T
    ii. 22 = W
    iii. 5 = F
    iv. 46 = u