Problem Statement 2:

Branching and Merging Create a feature branch for a specific enhancement in the project. Make changes in the feature branch and merge it back into the main branch. Resolve any merge conflicts that may arise during the merging process.

Solution :

Create a new branch named feature1 and push to git

```
C:\Users\DELL\simple website>git branch feature1

C:\Users\DELL\simple website>git branch
  feature1
* master
```

```
C:\Users\DELL\simple website>git push origin feature1
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 703 bytes | 234.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:      https://github.com/kavinvelavan/simple-website/pull/new/feature1
remote:
To https://github.com/kavinvelavan/simple-website.git
 * [new branch]      feature1 -> feature1
```

Make changes in feature and add to git

```
C:\Users\DELL\simple website>git branch
* feature1
  master

C:\Users\DELL\simple website>git add .

C:\Users\DELL\simple website>git commit -m "2 updated"
[feature1 982811b] 2 updated
 1 file changed, 4 insertions(+), 4 deletions(-)

C:\Users\DELL\simple website>git log --oneline
982811b (HEAD -> feature1) 2 updated
d1b3c69 (origin/feature1) first
2226b86 (origin/master, master) index

C:\Users\DELL\simple website>git push origin feature1
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 379 bytes | 126.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/kavinvelavan/simple-website.git
   d1b3c69..982811b  feature1 -> feature1
```

Merge the new feature to the main branch

The git merge command in Git is used to combine changes from different branches. When you merge one branch into another, Git incorporates the changes made in the source branch into the target branch.

```
C:\Users\DELL\simple website>git log --oneline
982811b (HEAD -> feature1, origin/feature1) 2 updated
d1b3c69 first
2226b86 (origin/master, master) index

C:\Users\DELL\simple website>git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

C:\Users\DELL\simple website>git pull origin master
From https://github.com/kavinvelavan/simple-website
 * branch            master     -> FETCH_HEAD
Already up to date.

C:\Users\DELL\simple website>git merge feature1
Updating 2226b86..982811b
Fast-forward
 src/firstpage.html | 41 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 41 insertions(+)
 create mode 100644 src/firstpage.html
```

Resolve any merge conflicts that may arise during the merging process.

When you are merging branches in Git, conflicts can occur if changes in the branches you are merging have affected the same lines of code. Resolving merge conflicts involves manually editing the conflicted files to incorporate the changes correctly.

1. Initiate the Merge:

git merge feature1

2. Conflicts Occur:

If there are conflicts, Git will pause the merge process and indicate which files have conflicts. You'll see a message like:

Auto-merging filename
CONFLICT (content): Merge conflict in filename

Automatic merge failed; fix conflicts and then commit the result.

3. Open Conflicted Files:

Open the conflicted file(s) in your code editor. Within these files, Git marks the conflicting sections:

```
    <<<< HEAD
# Changes from the current branch (main)
=======
# Changes from the branch being merged (feature-branch)
>>>>>>> feature1
```

4. Resolve Conflicts:

Manually edit the conflicted file to merge the changes correctly. You can choose to keep the changes from the current branch (HEAD), the branch being merged, or a combination of both.

Here's an example of how you might resolve a conflict in a file:

```
<<<<<<< HEAD
// Code from main branch
console.log("Hello from main branch");
=======
// Code from feature-branch
console.log("Hello from feature-branch");
>>>>>>> feature1
```

You can choose to keep one or both of the changes, or make entirely new changes. The resolved code might look like:

```
console.log("Hello from main branch and feature1");
```

5. Mark Conflicts as Resolved:

After resolving conflicts in a file, mark it as resolved using:

git add conflicted_file

6. Continue the Merge Process:

Continue the merge process:

git merge --continue

If you had more conflicts, Git will pause again for each conflicted file. Repeat steps 3-5 for each conflicted file until all conflicts are resolved.

7. Complete the Merge:

After resolving all conflicts and staging the changes, complete the merge:

git merge --continue

8. Commit the Merge:

If the merge completes successfully, commit the changes:

git commit -m "Merge feature1 into main"
git merge --abort

Resolving conflicts can be a manual and iterative process. After resolving conflicts, thoroughly test your code to ensure that the changes from both branches are integrated correctly.