# IT-314 Software Engineering
# Lab 7

**Code inspection, debugging and static analysis tool**

Hemal Ravrani

202201235

## 1. Program Inspection/Debugging for Long-code from GitHub

**We are given the following checklist and we have to find all the possible errors accordingly,**

1. Data referencing Errors
2. Data declaration Errors
3. Computation Errors
4. Comparison Errors
5. Control Flow errors
6. Interface errors
7. Input/Output Errors
8. Other Check

**First code from:** [**https://github.com/apache/echarts/blob/master/build/source-release/prepareReleaseMaterials.js**](https://github.com/apache/echarts/blob/master/build/source-release/prepareReleaseMaterials.js)

```
/*

* Licensed to the Apache Software Foundation (ASF) under one

* or more contributor license agreements.  See the NOTICE file

* distributed with this work for additional information

* regarding copyright ownership.  The ASF licenses this file
```

```
// Prepare release materials like mail, release note

const commander = require('commander');
const fse = require('fs-extra');
const pathTool = require('path');
const https = require('https');
```

```
commander
    .usage('[options]')
    .description([
        'Generate source release'
    ].join('\n'))
    .option(
        '--rcversion <version>',
        'Release version'
    )
    .option(
        '--commit <commit>',
        'Hash of commit'
    )
    .option(
        '--repo <repo>',
        'Repo'
    )
    .option(
        '--out <out>',
        'Out directory. Default to be tmp/release-mail'
    )
    .parse(process.argv);

const outDir = pathTool.resolve(process.cwd(),
commander.out || 'tmp/release-materials');
const releaseCommit = commander.commit;
if (!releaseCommit) {
    throw new Error('Release commit is required');
}
const repo = commander.repo || 'apache/echarts';

let rcVersion = commander.rcversion + '';
if (rcVersion.startsWith('v')) {   // tag may have v prefix, v5.1.0
    rcVersion = rcVersion.substr(1);
}
if (rcVersion.indexOf('-rc.') < 0) {
    throw new Error('Only rc version is accepeted.');
}

const parts = /(\d+)\.(\d+)\.(\d+)\-rc\.(\d+)/.exec(rcVersion);
if (!parts) {
    throw new Error(`Invalid version number ${rcVersion}`);
}
```

```
const major = +parts[1];
const minor = +parts[2];
const patch = +parts[3];
const rc = +parts[4];

const stableVersion = `${major}.${minor}.${patch}`;
const releaseFullName = `Apache ECharts ${stableVersion}
(release candidate ${rc})`;

console.log('[Release Repo] ' + repo);
console.log('[Release Verion] ' + rcVersion);
console.log('[Release Commit] ' + releaseCommit);
console.log('[Release Name] ' + releaseFullName);

const voteTpl = fse.readFileSync(pathTool.join(__dirname,
'./template/vote-release.tpl'), 'utf-8');
const voteResultTpl =
fse.readFileSync(pathTool.join(__dirname, './template/vote-
result.tpl'), 'utf-8');
const announceTpl =
fse.readFileSync(pathTool.join(__dirname,
'./template/announce-release.tpl'), 'utf-8');
```

```
const voteUntil = new Date(+new Date() + (72 + 12) * 3600 *
1000);   // 3.5 day.

fse.ensureDirSync(outDir);
fse.writeFileSync(
   pathTool.resolve(outDir, 'vote.txt'),
   voteTpl.replace(/{{ECHARTS_RELEASE_VERSION}}/g,
rcVersion)

.replace(/{{ECHARTS_RELEASE_VERSION_FULL_NAME}}/g,
releaseFullName)
      .replace(/{{ECHARTS_RELEASE_COMMIT}}/g,
releaseCommit)
      .replace(/{{VOTE_UNTIL}}/g, voteUntil.toISOString()),
   'utf-8'
);

fse.ensureDirSync(outDir);
fse.writeFileSync(
   pathTool.resolve(outDir, 'vote-result.txt'),
   voteResultTpl.replace(/{{ECHARTS_RELEASE_VERSION}}/g,
rcVersion)
```

```
.replace(/{{ECHARTS_RELEASE_VERSION_FULL_NAME}}/g,
releaseFullName),
    'utf-8'
);

fse.writeFileSync(
    pathTool.resolve(outDir, 'announce.txt'),
    announceTpl.replace(/{{ECHARTS_RELEASE_VERSION}}/g,
stableVersion)
        .replace(/{{ECHARTS_RELEASE_COMMIT}}/g,
releaseCommit),
    'utf-8'
);


// Fetch RELEASE_NOTE
https.get({
    hostname: 'api.github.com',
    path: `/repos/${repo}/releases`,
    headers: {
        'User-Agent': 'NodeJS'
    }
```

```
}, function (res) {

console.log(`https://api.github.com/repos/${repo}/releases`);
   if (res.statusCode !== 200) {
      console.error(`Failed to fetch releases
${res.statusCode}`);
      res.resume();
      return;
   }

   res.setEncoding('utf8');
   let rawData = '';
   res.on('data', (chunk) => {
      rawData += chunk;
   });
   res.on('end', () => {
      let releaseNote = '';

      const releases = JSON.parse(rawData);
      const found = releases.find(release => release.name ===
rcVersion);
      if (!found) {
         throw 'Can\'t found release';
      }
      else {
          releaseNote = found.body.trim();
          if (!releaseNote) {
              throw 'Release description is empty';
          }
      }

      const firstLine = releaseNote.split('\n')[0];
      if (firstLine.indexOf(stableVersion) < 0) {
          // Add version if release note is not start with version.
      }
      releaseNote = `## ${stableVersion}\n\n${releaseNote}`;

      fse.writeFileSync(
          pathTool.resolve(outDir, 'RELEASE_NOTE.txt'),
          releaseNote,
          'utf-8'
      );
  });
}).on('error', (e) => {
    throw e;
```

```
});
```

**1. Data Referencing Errors**

- **Error**: The script assumes that the provided repo follows the expected format. If an invalid repo is used, it may cause issues when fetching data from the GitHub API.

**2. Data Declaration Errors**

- **Error**: The rcVersion variable is derived from user input but is only partially validated. If rcVersion is not formatted correctly, it could lead to unexpected behavior.
- **Improvement Suggestion**: Ensure all variables have clear types and validate their content before use.

**3. Computation Errors**

- **Error**: The computation of voteUntil is based on a hardcoded number of hours (72 + 12). If there's a need for different durations in the future, it can lead to confusion or errors.

**4. Comparison Errors**

- **Error**: The check to see if rcVersion contains -rc. does not account for variations in spacing or capitalization. This may lead to missed validations.
- **Improvement Suggestion**: Use more robust string methods or regular expressions for comparison.

**5. Control Flow Errors**

- **Error**: The conditional statement for checking if the release note starts with the stable version does not implement any action when the condition is true. This may lead to incomplete or incorrect release notes.
- **Improvement Suggestion**: Implement logic to prepend the stable version to the release note if necessary.

**6. Interface Errors**

- **Error**: The command-line interface does not provide feedback for all options. If a required option is missing, it throws an error without providing clear guidance on how to correct it.

- **Improvement Suggestion**: Enhance user prompts and error messages to guide users in fixing input errors.

**7. Input/Output Errors**

- **Error**: The synchronous file operations (fse.readFileSync, fse.writeFileSync) may lead to blocking behavior, which can affect performance, especially in high-load scenarios.
- **Improvement Suggestion**: Use asynchronous methods for reading and writing files to prevent blocking the event loop.

**8. Other Checks**

- **Magic Numbers**: The script uses hardcoded values, like 72 + 12, without explanation. This practice can lead to confusion.
- **Documentation**: The lack of comments for complex logic sections makes it hard for others (or future you) to understand the intent behind the code.
- **Function Modularity**: The script could benefit from being broken down into smaller functions, improving readability and maintainability.

**Second Code:**
**https://github.com/apache/echarts/blob/master/build/addHeader.js**

```
/*
* Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements.  See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership.  The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License.  You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
```

```
* Unless required by applicable law or agreed to in writing,
* software distributed under the License is distributed on an
* "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
* KIND, either express or implied.  See the License for the
* specific language governing permissions and limitations
* under the License.
*/



const fs = require('fs');
const preamble = require('./preamble');
const pathTool = require('path');
const chalk = require('chalk');

// In the `.headerignore`, each line is a pattern in RegExp.
// all relative path (based on the echarts base directory) is tested.
// The pattern should match the relative path completely.
const excludesPath = pathTool.join(__dirname, '../.headerignore');
const ecBasePath = pathTool.join(__dirname, '../');

const isVerbose = process.argv[2] === '--verbose';

// const lists = [
//     '../src/**/*.js',
//     '../build/*.js',
//     '../benchmark/src/*.js',
//     '../benchmark/src/gulpfile.js',
//     '../extension-src/**/*.js',
//     '../extension/**/*.js',
//     '../map/js/**/*.js',
//     '../test/build/**/*.js',
//     '../test/node/**/*.js',
//     '../test/ut/core/*.js',
//     '../test/ut/spe/*.js',
//     '../test/ut/ut.js',
//     '../test/*.js',
//     '../theme/*.js',
//     '../theme/tool/**/*.js',
//     '../echarts.all.js',
//     '../echarts.blank.js',
//     '../echarts.common.js',
//     '../echarts.simple.js',
//     '../index.js',
//     '../index.common.js',
//     '../index.simple.js'
// ];

function run() {
    const updatedFiles = [];
    const passFiles = [];
    const pendingFiles = [];
```

```
  eachFile(function (absolutePath, fileExt) {
    const fileStr = fs.readFileSync(absolutePath, 'utf-8');

    const existLicense = preamble.extractLicense(fileStr, fileExt);

    if (existLicense) {
      passFiles.push(absolutePath);
      return;
    }

    // Conside binary files, only add for files with known ext.
    if (!preamble.hasPreamble(fileExt)) {
      pendingFiles.push(absolutePath);
      return;
    }

    fs.writeFileSync(absolutePath, preamble.addPreamble(fileStr, fileExt), 'utf-
8');
    updatedFiles.push(absolutePath);
  });

  console.log('\n');
  console.log('-----------------------------');
  console.log(' Files that exists license: ');
  console.log('-----------------------------');
  if (passFiles.length) {
    if (isVerbose) {
            passFiles.forEach(function (path) {
                console.log(chalk.green(path));
            });
        }
        else {
            console.log(chalk.green(passFiles.length + ' files. (use argument "--
verbose" see details)'));
        }
    }
    else {
        console.log('Nothing.');
    }

    console.log('\n');
    console.log('--------------------');
    console.log(' License added for: ');
    console.log('--------------------');
    if (updatedFiles.length) {
        updatedFiles.forEach(function (path) {
            console.log(chalk.green(path));
        });
    }
    else {
        console.log('Nothing.');
    }

    console.log('\n');
    console.log('----------------');
    console.log(' Pending files: ');
    console.log('----------------');
    if (pendingFiles.length) {
        pendingFiles.forEach(function (path) {
            console.log(chalk.red(path));
        });
    }
    else {
        console.log('Nothing.');
    }

    console.log('\nDone.');
}

function eachFile(visit) {

    const excludePatterns = [];
    const extReg = /\.([a-zA-Z0-9_-]+)$/;

    prepareExcludePatterns();
    travel('./');

    function travel(relativePath) {
        if (isExclude(relativePath)) {
            return;
        }

        const absolutePath = pathTool.join(ecBasePath, relativePath);
        const stat = fs.statSync(absolutePath);

        if (stat.isFile()) {
            visit(absolutePath, getExt(absolutePath));
        }
        else if (stat.isDirectory()) {
            fs.readdirSync(relativePath).forEach(function (file) {
                travel(pathTool.join(relativePath, file));
            });
        }
    }

    function prepareExcludePatterns() {
        const content = fs.readFileSync(excludesPath, {encoding: 'utf-8'});
        content.replace(/\r/g, '\n').split('\n').forEach(function (line) {
            line = line.trim();
            if (line && line.charAt(0) !== '#') {
                excludePatterns.push(new RegExp(line));
            }
        });
    }

    function isExclude(relativePath) {
        for (let i = 0; i < excludePatterns.length; i++) {
            if (excludePatterns[i].test(relativePath)) {
                return true;
            }
        }
    }

    function getExt(path) {
        if (path) {
            const mathResult = path.match(extReg);
            return mathResult && mathResult[1];
        }
    }
}

run();
```

**1. Data Referencing Errors**

- **Error**: The code assumes that the .headerignore file exists and is accessible. If the file does not exist or there are permission issues, it will throw an error.

**2. Data Declaration Errors**

- **Error**: The variable isVerbose relies on command-line arguments without validation. If the argument is not provided or malformed, it might lead to incorrect behavior.
- **Improvement Suggestion**: Validate command-line arguments more rigorously to ensure they conform to expected formats.

**3. Computation Errors**

- **Error**: The use of preamble.extractLicense and preamble.addPreamble methods assumes that the preamble module is correctly implemented. If there are bugs in those methods, it could lead to incorrect license processing.

**4. Comparison Errors**

- **Error**: The isExclude function uses regular expressions for path exclusion but does not handle edge cases effectively. If a path is partially matched by an exclusion pattern, it could lead to unintended behavior.
- **Improvement Suggestion**: Implement stricter checks or logging for excluded paths to understand why files may be skipped.

**5. Control Flow Errors**

- **Error**: The script does not handle the case where the fs.statSync method might throw an error if the path is inaccessible or not a directory/file. This can lead to unhandled exceptions.
- **Improvement Suggestion**: Implement try-catch blocks around filesystem operations to handle potential errors gracefully.

**6. Interface Errors**

- **Error**: The command-line interface does not provide enough feedback for users in case of errors or warnings. If no files are found or processed, the user may be left wondering what happened.
- **Improvement Suggestion**: Enhance user feedback for various scenarios (e.g., no files found, issues reading files) for better usability.

**7. Input/Output Errors**

- **Error**: The synchronous use of fs.readFileSync and fs.writeFileSync can lead to blocking behavior, which can impact performance when processing a large number of files.
- **Improvement Suggestion**: Consider using asynchronous file operations to improve performance and responsiveness.

**8. Other Checks**

- **Magic Numbers**: There are no magic numbers in this script, but the absence of documentation for hardcoded paths may lead to confusion.

- **Documentation**: There is a lack of comments explaining the purpose of functions or complex logic, which can make maintenance difficult.
- **Function Modularity**: While the code has some modular functions, the eachFile function could be further divided for better clarity and maintainability.

**Next: [https://github.com/apache/echarts/blob/master/build/build.js](https://github.com/apache/echarts/blob/master/build/build.js)**

```
#!/usr/bin/env node


/*
* Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements.  See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership.  The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License.  You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing,
* software distributed under the License is distributed on an
* "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
* KIND, either express or implied.  See the License for the
* specific language governing permissions and limitations
* under the License.
*/
```

```
const fs = require('fs');
const config = require('./config.js');
const commander = require('commander');
const chalk = require('chalk');
const rollup = require('rollup');
const prePublish = require('./pre-publish');
const transformDEV = require('./transform-dev');

async function run() {

    /**
     * Tips for `commander`:
     * (1) If arg xxx not specified, `commander.xxx` is undefined.
     *     Otherwise:
     *      If '-x, --xxx', `commander.xxx` can only be true/false, even if '--xxx yyy'
input.
     *      If '-x, --xxx <some>', the 'some' string is required, or otherwise error will
be thrown.
     *      If '-x, --xxx [some]', the 'some' string is optional, that is,
`commander.xxx` can be boolean or string.
     * (2) `node ./build/build.js --help` will print helper info and exit.
     */

    let descIndent = '                                 ';
    let egIndent = '    ';
```

```
commander
    .usage('[options]')
    .description([
        'Build echarts and generate result files in directory `echarts/dist`.',
        '',
        '  For example:',
        '',
        egIndent + 'node build/build.js --prepublish'
            + '\n' + descIndent + '# Only prepublish.',
        egIndent + 'node build/build.js --type ""'
            + '\n' + descIndent + '# Only generate `dist/echarts.js`.',
        egIndent + 'node build/build.js --type common --min'
            + '\n' + descIndent + '# Only generate `dist/echarts.common.min.js`.',
        egIndent + 'node build/build.js --type simple --min'
            + '\n' + descIndent + '# Only generate `dist/echarts-en.simple.min.js`.',
    ].join('\n'))
    .option(
        '--prepublish',
        'Build all for release'
    )
    .option(
        '--min',
        'Whether to compress the output file, and remove error-log-print code.'
    )
    .option(
        '--type <type name>', [
            'Can be "simple" or "common" or "all" (default). Or can be
simple,common,all to build multiple. For example,',
            descIndent + '`--type ""` or `--type "common"`.'
        ].join('\n'))
        .option(
            '--format <format>',
            'The format of output bundle. Can be "umd", "amd", "iife", "cjs", "esm".'
        )
        .parse(process.argv);

    let isPrePublish = !!commander.prepublish;
    let buildType = commander.type || 'all';

    let opt = {
        min: commander.min,
        format: commander.format || 'umd'
    };

    validateIO(opt.input, opt.output);

    if (isPrePublish) {
        await prePublish();
    }
    else if (buildType === 'extension') {
        const cfgs = [
            config.createBMap(opt),
            config.createDataTool(opt)
        ];
        await build(cfgs);
    }
    else if (buildType === 'ssr') {
        const cfgs = [
            config.createSSRClient(opt)
        ];
        await build(cfgs);
    }
    else if (buildType === 'myTransform') {
        const cfgs = [
            config.createMyTransform(opt)
        ];
        await build(cfgs);
    }
    else {
        const types = buildType.split(',').map(a => a.trim());


        // Since 5.5.0, echarts/package.json added `{"type": "module"}`, and
added
        // echarts/dist/package.json with `{"type": "commonjs"}`, both of which
makes
        // echarts/dist/echarts.esm.js can not be recognized as esm any more (at
least
        // in webpack5 and nodejs) any more. So we provides
echarts/dist/echarts.esm.mjs.
```

```
        // But for backward compat, we still provide provides
echarts/dist/echarts.esm.js.
        const isBuildingDistESM = (opt.format || '').toLowerCase() === 'esm';
        if (isBuildingDistESM) {
            await makeConfigAndBuild(opt, '.js');
            await makeConfigAndBuild(opt, '.mjs');
        }
        else {
            await makeConfigAndBuild(opt);
        }

        async function makeConfigAndBuild(opt, fileExtension) {
            const cfgs = types.map(type =>
                config.createECharts({
                    ...opt,
                    type,
                    fileExtension
                })
            );
            await build(cfgs);
        }
    }
}

function checkBundleCode(cfg) {
    // Make sure process.env.NODE_ENV is eliminated.
    for (let output of cfg.output) {
        let code = fs.readFileSync(output.file, {encoding: 'utf-8'});
        if (!code) {
            throw new Error(`${output.file} is empty`);
        }
        transformDEV.recheckDEV(code);
        console.log(chalk.green.dim('Check code: correct.'));
    }
}

function validateIO(input, output) {
    if ((input != null && output == null)
        || (input == null && output != null)
    ) {
        throw new Error('`input` and `output` must be both set.');
    }
}

/**
 * @param {Array.<Object>} configs A list of rollup configs:
 *  See: <https://rollupjs.org/#big-list-of-options>
 *  For example:
 *  [
 *     {
 *         ...inputOptions,
 *         output: [outputOptions],
 *     },
 *     ...
 *  ]
 */
async function build(configs) {
   console.log(chalk.yellow(`
   NOTICE: If you are using 'npm run build'. Run 'npm run prepare' before build
!!!
`));

   console.log(chalk.yellow(`
   NOTICE: If you are using syslink on zrender. Run 'npm run prepare' in zrender
first !!
`));

   for (let singleConfig of configs) {
      console.log(
         chalk.cyan.dim('\Bundling '),
         chalk.cyan(singleConfig.input)
      );

      console.time('rollup build');
      const bundle = await rollup.rollup(singleConfig);

      for (let output of singleConfig.output) {
         console.log(
            chalk.green.dim('Created '),
            chalk.green(output.file),
                chalk.green.dim(' successfully.')
            );

            await bundle.write(output);

        };
        console.timeEnd('rollup build');

        checkBundleCode(singleConfig);
    }
}

async function main() {
    try {
        await run();
    }
    catch (err) {
        console.log(chalk.red('BUILD ERROR!'));
        // rollup parse error.
        if (err) {
            if (err.loc) {
                console.warn(chalk.red(`${err.loc.file}
(${err.loc.line}:${err.loc.column})`));
                console.warn(chalk.red(err.message));
            }
            if (err.frame) {
                console.warn(chalk.red(err.frame));
            }
            console.log(chalk.red(err ? err.stack : err));

            err.id != null && console.warn(chalk.red(`id: ${err.id}`));
            err.hook != null && console.warn(chalk.red(`hook: ${err.hook}`));
            err.code != null && console.warn(chalk.red(`code: ${err.code}`));
            err.plugin != null && console.warn(chalk.red(`plugin: ${err.plugin}`));
        }
        // console.log(err);
    }
}

main();
```

**1. Data Referencing Errors**

- **Error**: The validateIO function checks if input and output are both set, but these variables are never initialized in the run function. This will lead to undefined values being checked, potentially resulting in unexpected behavior.

**2. Data Declaration Errors**

- **Error**: The opt variable is created with properties min and format, but input and output are referenced before being assigned any values. This could lead to errors in the build process if the options are not properly defined.
- **Improvement Suggestion**: Ensure input and output are declared and initialized before being passed to validateIO.

**3. Computation Errors**

- **Error**: If the rollup build fails for any reason, the error handling in main may not catch specific build errors cleanly. The process might not terminate correctly, leading to inconsistent build states.

- **Improvement Suggestion**: Implement more granular error handling for various stages of the rollup build to identify where the failure occurs.

**4. Comparison Errors**

- **Error**: In the validateIO function, the comparison of input and output for null values is done using !=. This is generally acceptable, but using strict equality (!==) would prevent unintended type coercion.
- **Improvement Suggestion**: Use strict equality checks to avoid potential bugs related to type coercion.

**5. Control Flow Errors**

- **Error**: The checkBundleCode function assumes that cfg.output will always have valid paths. If output.file is not set correctly, it could throw an error when trying to read a non-existent file.
- **Improvement Suggestion**: Add checks to ensure the output file exists before attempting to read it.

**6. Interface Errors**

- **Error**: The commander options are quite verbose, and users may find it difficult to understand the command usage without adequate inline help or examples.
- **Improvement Suggestion**: Provide clearer descriptions or help for each command-line option to enhance usability.

**7. Input/Output Errors**

- **Error**: The script reads from the filesystem synchronously in several places (e.g., fs.readFileSync), which can block the event loop and affect performance when processing large files or numerous files.
- **Improvement Suggestion**: Consider using asynchronous versions of file I/O functions to improve performance and responsiveness.

**8. Other Checks**

- **Error Handling**: The script has a general catch for errors, but it could improve the clarity and specificity of error messages to assist users in troubleshooting.

- **Magic Numbers**: There are no magic numbers, but the hardcoded output paths and configuration options could benefit from being extracted into constants for better maintainability.
- **Documentation**: The code could use more comments explaining the purpose of specific functions and logic, especially for complex parts of the build process.

**Next:** https://github.com/apache/echarts/blob/master/build/pre-publish.js

```
/*
* Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements.  See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership.  The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License.  You may obtain a copy of the License at
*
*   http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing,
* software distributed under the License is distributed on an
* "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
* KIND, either express or implied.  See the License for the
* specific language governing permissions and limitations
* under the License.
*/

/**
```

```
 * [Create CommonJS files]:
 * Compatible with previous folder structure: `echarts/lib` exists in
`node_modules`
 * (1) Build all files to CommonJS to `echarts/lib`.
 * (2) Remove __DEV__.
 * (3) Mount `echarts/src/export.js` to `echarts/lib/echarts.js`.
 *
 * [Create ESModule files]:
 * Build all files to CommonJS to `echarts/esm`.
 */

const nodePath = require('path');
const assert = require('assert');
const fs = require('fs');
const fsExtra = require('fs-extra');
const chalk = require('chalk');
const ts = require('typescript');
const globby = require('globby');
const transformDEVUtil = require('./transform-dev');
const preamble = require('./preamble');
const dts = require('@lang/rollup-plugin-dts').default;
const rollup = require('rollup');
const { transformImport } = require('zrender/build/transformImport.js');

const ecDir = nodePath.resolve(__dirname, '..');
const tmpDir = nodePath.resolve(ecDir, 'pre-publish-tmp');
const tsConfig = readTSConfig();

const autoGeneratedFileAlert = `
/**
 * AUTO-GENERATED FILE. DO NOT MODIFY.
 */

`;

const mainSrcGlobby = {
    patterns: [
        'src/**/*.ts'
    ],
    cwd: ecDir
};
const extensionSrcGlobby = {
    patterns: [
        'extension-src/**/*.ts'
    ],
    cwd: ecDir
};
const extensionSrcDir = nodePath.resolve(ecDir, 'extension-src');
const extensionESMDir = nodePath.resolve(ecDir, 'extension');
const ssrClientGlobby = {
    patterns: [
        'ssr/client/src/**/*.ts'
    ],
    cwd: ecDir
};
const ssrClientSrcDir = nodePath.resolve(ecDir, 'ssr/client/src');
const ssrClientESMDir = nodePath.resolve(ecDir, 'ssr/client/lib');
const ssrClientTypeDir = nodePath.resolve(ecDir, 'ssr/client/types');

const typesDir = nodePath.resolve(ecDir, 'types');
const esmDir = 'lib';



const compileWorkList = [
    {
        logLabel: 'main ts -> js-esm',
        compilerOptionsOverride: {
            module: 'ES2015',
            rootDir: ecDir,
            outDir: tmpDir,
            // Generate types when building esm
            declaration: true,
            declarationDir: typesDir
        },
        srcGlobby: mainSrcGlobby,
        transformOptions: {
            filesGlobby: {patterns: ['**/*.js'], cwd: tmpDir},
            preamble: preamble.js,
            transformDEV: true
        },
        before: async function () {
            fsExtra.removeSync(tmpDir);
            fsExtra.removeSync(nodePath.resolve(ecDir, 'types'));
            fsExtra.removeSync(nodePath.resolve(ecDir, esmDir));
            fsExtra.removeSync(nodePath.resolve(ecDir, 'index.js'));
            fsExtra.removeSync(nodePath.resolve(ecDir, 'index.blank.js'));
            fsExtra.removeSync(nodePath.resolve(ecDir, 'index.common.js'));
            fsExtra.removeSync(nodePath.resolve(ecDir, 'index.simple.js'));
        },
        after: async function () {
            fs.renameSync(nodePath.resolve(tmpDir, 'src/echarts.all.js'),
nodePath.resolve(ecDir, 'index.js'));
            fs.renameSync(nodePath.resolve(tmpDir, 'src/echarts.blank.js'),
nodePath.resolve(ecDir, 'index.blank.js'));
            fs.renameSync(nodePath.resolve(tmpDir, 'src/echarts.common.js'),
nodePath.resolve(ecDir, 'index.common.js'));
            fs.renameSync(nodePath.resolve(tmpDir, 'src/echarts.simple.js'),
nodePath.resolve(ecDir, 'index.simple.js'));
            fs.renameSync(nodePath.resolve(tmpDir, 'src'), nodePath.resolve(ecDir,
esmDir));

            transformRootFolderInEntry(nodePath.resolve(ecDir, 'index.js'), esmDir);
            transformRootFolderInEntry(nodePath.resolve(ecDir, 'index.blank.js'),
esmDir);
            transformRootFolderInEntry(nodePath.resolve(ecDir, 'index.common.js'),
esmDir);
```

```
            transformRootFolderInEntry(nodePath.resolve(ecDir, 'index.simple.js'),
esmDir);


            await transformLibFiles(nodePath.resolve(ecDir, esmDir), esmDir);
            await transformLibFiles(nodePath.resolve(ecDir, 'types'), esmDir);
            fsExtra.removeSync(tmpDir);
        }
    },
    {
        logLabel: 'extension ts -> js-esm',
        compilerOptionsOverride: {
            module: 'ES2015',
            declaration: false,
            rootDir: extensionSrcDir,
            outDir: extensionESMDir
        },
        srcGlobby: extensionSrcGlobby,
        transformOptions: {
            filesGlobby: {patterns: ['**/*.js'], cwd: extensionESMDir},
            preamble: preamble.js,
            transformDEV: true
        },
        before: async function () {
            fsExtra.removeSync(extensionESMDir);
        },
        after: async function () {
            await transformLibFiles(extensionESMDir, 'lib');
        }
    },
    {
        logLabel: 'ssr client ts -> js-esm',
        compilerOptionsOverride: {
            module: 'ES2015',
            declaration: true,
            rootDir: ssrClientSrcDir,
            outDir: ssrClientESMDir,
            declarationDir: ssrClientTypeDir
        },
        srcGlobby: ssrClientGlobby,
        transformOptions: {
            filesGlobby: {patterns: ['**/*.js'], cwd: ssrClientESMDir},
            transformDEV: true
        },
        before: async function () {
            fsExtra.removeSync(ssrClientESMDir);
        },
        after: async function () {
            await transformLibFiles(ssrClientESMDir, 'lib');
        }
    }
];
```

```
/**
 * @public
 */
module.exports = async function () {

   for (let {
      logLabel, compilerOptionsOverride, srcGlobby,
      transformOptions, before, after
   } of compileWorkList) {

      process.stdout.write(chalk.green.dim(`[${logLabel}]: compiling ...`));

      before && await before();

      let srcPathList = await readFilePaths(srcGlobby);

      await tsCompile(compilerOptionsOverride, srcPathList);

      process.stdout.write(chalk.green.dim(` done \n`));

      process.stdout.write(chalk.green.dim(`[${logLabel}]: transforming ...`));

      await transformCode(transformOptions);

      after && await after();
```

```
        process.stdout.write(chalk.green.dim(` done \n`));
    }

    process.stdout.write(chalk.green.dim(`Generating entries ...`));
    generateEntries();
    process.stdout.write(chalk.green.dim(`Bundling DTS ...`));
    await bundleDTS();

    console.log(chalk.green.dim('All done.'));
};

async function runTsCompile(localTs, compilerOptions, srcPathList) {
    // Must do it, because the value in tsconfig.json might be different from the
inner representation.
    // For example: moduleResolution: "NODE" => moduleResolution: 2
    const {options, errors} =
localTs.convertCompilerOptionsFromJson(compilerOptions, ecDir);

    if (errors.length) {
        let errMsg = 'tsconfig parse failed: '
            + errors.map(error => error.messageText).join('. ')
            + '\n compilerOptions: \n' + JSON.stringify(compilerOptions, null, 4);
        assert(false, errMsg);
    }
```

```
  // See: https://github.com/microsoft/TypeScript/wiki/Using-the-Compiler-
API

  let program = localTs.createProgram(srcPathList, options);
  let emitResult = program.emit();

  let allDiagnostics = localTs
    .getPreEmitDiagnostics(program)
    .concat(emitResult.diagnostics);

  allDiagnostics.forEach(diagnostic => {
    if (diagnostic.file) {
      let {line, character} =
diagnostic.file.getLineAndCharacterOfPosition(diagnostic.start);
      let message =
localTs.flattenDiagnosticMessageText(diagnostic.messageText, '\n');
      console.log(chalk.red(`${diagnostic.file.fileName} (${line + 1},${character
+ 1}): ${message}`));
    }
    else {

console.log(chalk.red(localTs.flattenDiagnosticMessageText(diagnostic.message
Text, '\n')));
    }
  });
  if (allDiagnostics.length > 0) {
    throw new Error('TypeScript Compile Failed')
  }
```

```
}
module.exports.runTsCompile = runTsCompile;


async function tsCompile(compilerOptionsOverride, srcPathList) {
   assert(
      compilerOptionsOverride
      && compilerOptionsOverride.module
      && compilerOptionsOverride.rootDir
      && compilerOptionsOverride.outDir
   );

   let compilerOptions = {
      ...tsConfig.compilerOptions,
      ...compilerOptionsOverride,
      sourceMap: false
   };

   runTsCompile(ts, compilerOptions, srcPathList);
}

/**
 * Transform import/require path in the entry file to `esm` or `lib`.
 */
function transformRootFolderInEntry(entryFile, replacement) {
   let code = fs.readFileSync(entryFile, 'utf-8');
   // Simple regex replacement
```

```
    // TODO More robust way?
    assert(
        !/(import\s+|from\s+|require\(\s*)["']\.\/echarts\./.test(code)
        && !/(import\s+|from\s+|require\(\s*)["']echarts\./.test(code),
        'Import echarts.xxx.ts is not supported.'
    );
    code = code.replace(/((import\s+|from\s+|require\(\s*)["'])\.\//g,
`$1./${replacement}/`);
    fs.writeFileSync(
        entryFile,
        // Also transform zrender.
        singleTransformImport(code, replacement),
        'utf-8'
    );
}

/**
 * Transform `zrender/src` to `zrender/lib` in all files
 */
async function transformLibFiles(rooltFolder, replacement) {
    const files = await readFilePaths({
        patterns: ['**/*.js', '**/*.d.ts'],
        cwd: rooltFolder
    });
    // Simple regex replacement
    // TODO More robust way?
    for (let fileName of files) {
        let code = fs.readFileSync(fileName, 'utf-8');
        code = singleTransformImport(code, replacement);
        // For lower ts version, not use import type
        // TODO Use https://github.com/sandersn/downlevel-dts ?
        // if (fileName.endsWith('.d.ts')) {
        //     code = singleTransformImportType(code);
        // }
        fs.writeFileSync(fileName, code, 'utf-8');
    }
}

/**
 * 1. Transform zrender/src to zrender/lib
 * 2. Add .js extensions
 */
function singleTransformImport(code, replacement) {
    return transformImport(
        code.replace(/([\"\'])zrender\/src\//g, `$1zrender/${replacement}/`),
        (moduleName) => {
            // Ignore 'tslib' and 'echarts' in the extensions.
            if (moduleName === 'tslib' || moduleName === 'echarts') {
                return moduleName;
            }
            else if (moduleName === 'zrender/lib/export') {
                throw new Error('Should not import the whole zrender library.');
            }
```

```
          else if (moduleName.endsWith('.ts')) {
             // Replace ts with js
             return moduleName.replace(/\.ts$/, '.js');
          }
          else if (moduleName.endsWith('.js')) {
             return moduleName;
          }
          else {
             return moduleName + '.js'
          }
       }
   );
}

// function singleTransformImportType(code) {
//     return code.replace(/import\s+type\s+/g, 'import ');
// }

/**
 * @param {Object} transformOptions
 * @param {Object} transformOptions.filesGlobby {patterns: string[], cwd: 
string}
 * @param {string} [transformOptions.preamble] See './preamble.js'
 * @param {boolean} [transformOptions.transformDEV]
 */
async function transformCode({filesGlobby, preamble, transformDEV}) {
   let filePaths = await readFilePaths(filesGlobby);

   filePaths.map(filePath => {
       let code = fs.readFileSync(filePath, 'utf8');

       if (transformDEV) {
          let result = transformDEVUtil.transform(code, false);
          code = result.code;
       }

       code = autoGeneratedFileAlert + code;

       if (preamble) {
          code = preamble + code;
       }

       fs.writeFileSync(filePath, code, 'utf8');
   });
}

async function readFilePaths({patterns, cwd}) {
   assert(patterns && cwd);
   return (
       await globby(patterns, {cwd})
   ).map(
       srcPath => nodePath.resolve(cwd, srcPath)
   );
}

// Bundle can be used in echarts-examples.
async function bundleDTS() {

   const outDir = nodePath.resolve(__dirname, '../types/dist');
   const commonConfig = {
       onwarn(warning, rollupWarn) {
          // Not warn circular dependency
          if (warning.code !== 'CIRCULAR_DEPENDENCY') {
              rollupWarn(warning);
          }
       },
       plugins: [
          dts({
              respectExternal: true
          })
//            {
//                generateBundle(options, bundle) {
//                    for (let chunk of Object.values(bundle)) {
//                        chunk.code = `
// type Omit<T, K> = Pick<T, Exclude<keyof T, K>>;
// ${chunk.code}`
//                    }
//                }
//          }
      ]
   };

   // Bundle chunks.
   const parts = [
      'core', 'charts', 'components', 'renderers', 'option', 'features'
   ];
   const inputs = {};
   parts.forEach(partName => {
      inputs[partName] = nodePath.resolve(__dirname,
`../types/src/export/${partName}.d.ts`)
   });

   const bundle = await rollup.rollup({
      input: inputs,
      ...commonConfig
   });
   let idx = 1;
   await bundle.write({
      dir: outDir,
      minifyInternalExports: false,
      manualChunks: (id) => {
         // Only create one chunk.
         return 'shared';
      },
      chunkFileNames: 'shared.d.ts'
```

```
    });

    // Bundle all in one
    const bundleAllInOne = await rollup.rollup({
        input: nodePath.resolve(__dirname, `../types/src/export/all.d.ts`),
        ...commonConfig
    });
    await bundleAllInOne.write({
        file: nodePath.resolve(outDir, 'echarts.d.ts')
    });
}

function readTSConfig() {
    // tsconfig.json may have comment string, which is invalid if
    // using `require('tsconfig.json'). So we use a loose parser.
    let filePath = nodePath.resolve(ecDir, 'tsconfig.json');
    const tsConfigText = fs.readFileSync(filePath, {encoding: 'utf8'});
    return (new Function(`return ( ${tsConfigText} )`))();
}


function generateEntries() {
    ['charts', 'components', 'renderers', 'core', 'features',
'ssr/client/index'].forEach(entryPath => {
        if (entryPath !== 'option') {
            const jsCode = fs.readFileSync(nodePath.join(__dirname,
`template/${entryPath}.js`), 'utf-8');
```

```
        fs.writeFileSync(nodePath.join(__dirname, `../${entryPath}.js`), jsCode,
'utf-8');
      }

      // Make the d.ts in the same dir as .js, so that the can be found by tsc.
      // package.json "types" in "exports" does not always seam to work.
      const dtsCode = fs.readFileSync(nodePath.join(__dirname,
`/template/${entryPath}.d.ts`), 'utf-8');
      fs.writeFileSync(nodePath.join(__dirname, `../${entryPath}.d.ts`), dtsCode,
'utf-8');
   });
}

module.exports.readTSConfig = readTSConfig;
```

**1. Data Referencing Errors**

These errors occur when the code tries to access variables or data structures that are not defined or initialized.

- **Undefined Variables**: Make sure all variables referenced in the code are defined. For example, ensure that preamble is correctly imported from preamble.js.
- **Check for Typos**: Verify that the variable names used in the code match exactly with their declarations.

**2. Data Declaration Errors**

These involve incorrect declarations or initializations of variables.

- **Incorrect Variable Types**: Ensure that variables are declared with the correct data types. For example, if a variable is expected to be an array or object, it should be initialized accordingly.
- **Use of let, const, var**: Make sure to use let or const for variable declarations to avoid scope-related issues.

**3. Computation Errors**

These errors arise during mathematical operations or function evaluations.

- **Arithmetic Operations**: Verify that all arithmetic operations are performed on compatible data types. For instance, operations between strings and numbers can lead to unexpected results.
- **Function Logic**: Ensure the logic within functions such as transformCode() is correct. Double-check any calculations or transformations for correctness.

**4. Comparison Errors**

These occur when comparisons are made between incompatible types or values.

- **Use of === vs ==**: Ensure that you are using strict equality (===) to avoid type coercion issues in comparisons.
- **Logical Operators**: Check that logical comparisons in conditions are valid and meaningful, especially in control flow statements.

**5. Control Flow Errors**

These errors relate to the flow of execution in the code.

- **Incorrect Use of Conditionals**: Ensure that if statements and loops have the correct conditions. Misconfigured conditions can lead to unintended branches in code execution.
- **Proper Looping Constructs**: Verify that loops (e.g., for, while) have valid terminating conditions to avoid infinite loops.

**6. Interface Errors**

These involve incorrect or unexpected interactions with external systems or APIs.

- **File System Operations**: Check that file read/write operations interact correctly with the filesystem. If a file does not exist, this can lead to runtime errors.
- **Invalid Function Calls**: Ensure that the functions being called (like readFilePaths(), rollup(), etc.) are invoked with the correct parameters and types.

**7. Input/Output Errors**

These errors occur when reading from or writing to external sources (files, databases, etc.).

- **File Existence**: Check that files referenced (e.g., from globby) actually exist before trying to read from or delete them.
- **Output Path Issues**: Ensure that the output paths are valid and writable. If you're trying to write to a directory that doesn't exist, it will throw an error.

**8. Other Checks**

These encompass various other potential issues in the code.

- **Logging and Error Handling**: Make sure to include error handling for critical operations (e.g., file I/O) and log helpful error messages for easier debugging.
- **Testing Edge Cases**: Test your code with various inputs to cover edge cases, ensuring that it behaves as expected under different conditions.
- **Version Compatibility**: Ensure that the libraries and modules used (like rollup, typescript, etc.) are compatible with your current environment.

**Suggested Improvements**

1. **Add Error Handling**: Wrap file I/O and external API calls in try-catch blocks to handle potential exceptions gracefully.
2. **Use Type Checking**: In TypeScript, leverage type annotations to catch type-related errors at compile time.
3. **Validate Inputs**: Before processing inputs, validate them to ensure they meet the expected criteria (e.g., not null, correct format).
4. **Test Thoroughly**: Create unit tests for critical functions to verify their behavior under different scenarios.

## [ii] CODE DEBUGGING:

**Debugging is the process of localizing, analysing, and removing suspected errors in the code (Java code given in the .zip file)**

### [1] Armstrong Number

Errors Identified:

1. Logic Error in the Remainder Calculation:
   - In the while(num > 0) loop, remainder = num / 10 is incorrect. The code divides the number by 10, which reduces it prematurely. The correct operation should be remainder = num % 10 to extract the last digit of the number.
2. Logic Error in Reducing num:
   - num = num % 10 is also incorrect because this operation extracts the last digit, but the purpose is to drop the last digit. The correct statement should be num = num / 10 to remove the last digit.

**Revised Code:**

```
// Armstrong Number
class Armstrong {
   public static void main(String args[]) {
      int num = Integer.parseInt(args[0]);
      int n = num; // used to check at last time
      int check = 0, remainder;
      while(num > 0) {
         remainder = num % 10; // Corrected: extracting the last digit
         check = check + (int)Math.pow(remainder, 3); // Corrected logic
         num = num / 10; // Corrected: removing the last digit
      }
      if(check == n)
         System.out.println(n + " is an Armstrong Number");
```

```
        else
            System.out.println(n + " is not an Armstrong Number");
    }
}
```

**Program Inspection Analysis:**

1. **How many errors are there in the program?**
    - There are 3 main errors:
        - Incorrect remainder calculation
        - Incorrect reduction of the number
2. **Which category of program inspection would you find more effective?**
    - **Control-Flow Errors (Category E)**: This helped in identifying the issues related to incorrect looping and the wrong calculations in extracting the digits.
    - **Computation Errors (Category C)**: For the errors in the calculations, such as how the Armstrong number check was failing.
3. **Which type of error were you not able to identify using the program inspection?**
    - The article in the output ("an Armstrong Number") would be hard to spot through program inspection unless a more detailed check for user-facing messages is performed.
4. **Is the program inspection technique worth applying?**
    - Yes, it helps catch both computational and control flow errors that might not immediately surface through regular testing, especially when dealing with numeric operations and loops.

**[2] GCD and LCM**

**Errors Identified:**

1. **GCD Function Logic Error:**

- In the gcd() method, the condition in the while loop is incorrect: while(a % b == 0). This will terminate the loop immediately if a is divisible by b, which is not the intended logic. The condition should be while(a % b != 0) to continue until the remainder is 0.
- Additionally, the variables a and b are swapped incorrectly. The correct interpretation is that a should be the larger number and b the smaller.

2. **LCM Function Logic Error:**
    - In the lcm() method, the condition if(a % x != 0 && a % y != 0) is incorrect. The correct logic is if(a % x == 0 && a % y == 0), as we want to find the smallest a that is divisible by both x and y.
3. **Output Error:**
    - The second output message mistakenly says "The GCD of two numbers is: 20" instead of "The LCM of two numbers is: 20." This is a typo in the output message.

**Revised Code:**

```
import java.util.Scanner;


public class GCD_LCM

{

  static int gcd(int x, int y)

  {

    int r = 0, a, b;

    a = (x > y) ? x : y; // a is the larger number

    b = (x < y) ? x : y; // b is the smaller number


    while(b != 0) // Corrected: Continue loop until b becomes 0

    {

      r = a % b;
```

```
            a = b;
            b = r;
        }
        return a;
    }

    static int lcm(int x, int y)
    {
        int a = (x > y) ? x : y; // a is the greater number
        while(true)
        {
            if(a % x == 0 && a % y == 0) // Corrected: find LCM
                return a;
            ++a;
        }
    }

    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the two numbers: ");
        int x = input.nextInt();
        int y = input.nextInt();

        System.out.println("The GCD of two numbers is: " + gcd(x, y));
        System.out.println("The LCM of two numbers is: " + lcm(x, y)); //
Corrected the output text
```

```
        input.close();
    }
}
```

**Program Inspection Analysis:**

1. **How many errors are there in the program?**
   - There are 3 main errors:
     - The incorrect loop condition in the GCD function.
     - Incorrect condition for finding the LCM.
     - Typo in the output message for LCM.
2. **Which category of program inspection would you find more effective?**
   - **Control-Flow Errors (Category E)**: Helped identify the incorrect loop condition in both the GCD and LCM functions.
   - **Computation Errors (Category C)**: For catching the logical issues in both the GCD and LCM calculations.
3. **Which type of error were you not able to identify using program inspection?**
   - The output typo in the LCM message could have been overlooked during program inspection.
4. **Is the program inspection technique worth applying?**
   - Yes, especially for finding control-flow errors in the loop and logic errors in calculations. It helps ensure the program behaves as intended.

**[3] Knapsack**

**Errors Identified:**

1. **Logical Error in option1 Calculation:**
   - option1 is calculated as opt[n++][w], which increments n in place. This is incorrect as n++ modifies n prematurely. The correct code should just access the current value without incrementing it, i.e., opt[n][w].

2. **Logical Error in option2 Calculation:**
    - option2 = profit[n-2] + opt[n-1][w-weight[n]] is incorrect. The subtraction profit[n-2] seems to be a mistake as it references the wrong item. It should be profit[n] (the current item's profit).
3. **Invalid Weight Check Condition:**
    - The condition if (weight[n] > w) should return option2 = Integer.MIN_VALUE if the item's weight exceeds the current limit. This part is actually reversed and should be: if (weight[n] <= w) so that we calculate option2 correctly when the weight is within bounds.
4. **Array Bound Issues:**
    - There might be array out-of-bounds errors if weight[n] > w. The boundary conditions need to be carefully checked.
5. **Output Formatting Error:**
    - The output column headers include "Item" but the output lists "item". A minor consistency issue here.

**Corrected Code:**

```
public class Knapsack {


  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);   // number of items
    int W = Integer.parseInt(args[1]);   // maximum weight of knapsack


    int[] profit = new int[N+1];
    int[] weight = new int[N+1];


    // generate random instance, items 1..N
    for (int n = 1; n <= N; n++) {
      profit[n] = (int) (Math.random() * 1000);
```

```
          weight[n] = (int) (Math.random() * W);
        }

        // opt[n][w] = max profit of packing items 1..n with weight limit w
        // sol[n][w] = does the optimal solution to pack items 1..n with weight
limit w include item n?
        int[][] opt = new int[N+1][W+1];
        boolean[][] sol = new boolean[N+1][W+1];

        // Fill in the knapsack table
        for (int n = 1; n <= N; n++) {
          for (int w = 1; w <= W; w++) {

            // don't take item n
            int option1 = opt[n-1][w];

            // take item n, if it fits
            int option2 = Integer.MIN_VALUE;
            if (weight[n] <= w)
              option2 = profit[n] + opt[n-1][w - weight[n]];

            // select better of two options
            opt[n][w] = Math.max(option1, option2);
            sol[n][w] = (option2 > option1);
          }
        }
        // determine which items to take
        boolean[] take = new boolean[N+1];
        for (int n = N, w = W; n > 0; n--) {
          if (sol[n][w]) {
            take[n] = true;
            w = w - weight[n];
          }
          else {
            take[n] = false;
          }
        }

        // print results
        System.out.println("Item" + "\t" + "Profit" + "\t" + "Weight" + "\t" +
"Take");
        for (int n = 1; n <= N; n++) {
          System.out.println(n + "\t" + profit[n] + "\t" + weight[n] + "\t" +
take[n]);
        }
    }
}
```

**Program Inspection Summary:**

1. **Errors Found**: 3 main logical errors related to increment/decrement misuse, condition check errors, and array reference issues.
2. **Effective Techniques**: **Control Flow Error (Category E)** to fix loop conditions and proper use of array bounds. **Computation Errors (Category C)** for fixing profit calculation logic.

3. **Unidentified Errors**: Without inspecting the code, it might be harder to catch these logical issues since they result in incorrect behavior rather than runtime exceptions.

**[4] Magic Number**

**Key Errors:**

1. **Error in summing digits:** The logic for summing digits inside the inner while loop is incorrect. The operation s = s * (sum / 10) is not correct for summing digits. It should instead be adding the last digit of the number.
2. **While loop condition:** The condition while (sum == 0) is incorrect; it should be while (sum > 0) to continue summing digits while sum has any digits left.
3. **Missing semicolon:** The statement sum = sum % 10 is missing a semicolon.

**Corrected Code:**

```
import java.util.*;


public class MagicNumberCheck {
  public static void main(String args[]) {
    Scanner ob = new Scanner(System.in);
    System.out.println("Enter the number to be checked.");
    int n = ob.nextInt();
    int sum = 0, num = n;


    // Repeat the process of summing the digits until the number becomes a single digit
    while (num > 9) {
      sum = num;
      int s = 0;
```

```
            // Sum the digits of the number
            while (sum > 0) {
                s += sum % 10;  // Add the last digit to 's'
                sum /= 10;      // Remove the last digit from 'sum'
            }
            num = s;  // Update num to the sum of digits
        }

        // Check if the final number is 1 (magic number)
        if (num == 1) {
            System.out.println(n + " is a Magic Number.");
        } else {
            System.out.println(n + " is not a Magic Number.");
        }

        ob.close();  // Close the scanner to avoid resource leaks
    }
}
```

**Explanation of Changes:**

**Fixed logic for summing digits**: The inner loop now correctly sums the digits using s += sum % 10 and reduces the number with sum /= 10.

**Corrected loop condition**: The condition inside the digit-summing loop is changed to while (sum > 0) to ensure it runs until all digits are processed.

**Added missing semicolon:** Corrected the syntax errors by adding the semicolon in the sum = sum % 10 line.

**Closing the scanner**: Added ob.close() to avoid resource leaks.

**Program Inspection Summary:**

- **Control Flow Error (Category E):** Incorrect loop condition and operations were addressed.
- **Computation Errors (Category C):** Fixed the calculation of digit sums.

**[5] Merge Sort**

**Issues:**

1. **Array slicing errors in mergeSort:**
   - The code uses array+1 and array-1 to slice arrays, which is invalid. Arrays in Java cannot be modified like that; proper methods need to be used for splitting them.
2. **Incorrect increment/decrement operations:**
   - The code uses left++ and right-- when passing arrays to the merge function. These are not valid operations on arrays.

**Revised Code:**

```
import java.util.*;


public class MergeSort {
  public static void main(String[] args) {
    int[] list = {14, 32, 67, 76, 23, 41, 58, 85};
    System.out.println("before: " + Arrays.toString(list));
    mergeSort(list);
    System.out.println("after:  " + Arrays.toString(list));
  }


  // Sorts the given array using the merge sort algorithm.
  public static void mergeSort(int[] array) {
    if (array.length > 1) {
      // split the array into two halves
      int[] left = leftHalf(array);
      int[] right = rightHalf(array);

      // recursively sort the two halves
      mergeSort(left);
      mergeSort(right);

      // merge the sorted halves into a sorted whole
      merge(array, left, right);
    }
  }

  // Returns the first half of the given array.
  public static int[] leftHalf(int[] array) {
    int size1 = array.length / 2;
    int[] left = new int[size1];
    for (int i = 0; i < size1; i++) {
      left[i] = array[i];
    }
    return left;
  }

  // Returns the second half of the given array.
  public static int[] rightHalf(int[] array) {
    int size1 = array.length / 2;
    int size2 = array.length - size1;
    int[] right = new int[size2];
```

```
        for (int i = 0; i < size2; i++) {
            right[i] = array[i + size1];
        }
        return right;
    }

    // Merges the given left and right arrays into the given result array.
    public static void merge(int[] result, int[] left, int[] right) {
        int i1 = 0;  // index into left array
        int i2 = 0;  // index into right array

        for (int i = 0; i < result.length; i++) {
            if (i2 >= right.length || (i1 < left.length && left[i1] <= right[i2])) {
                result[i] = left[i1];  // take from left
                i1++;
            } else {
                result[i] = right[i2];  // take from right
                i2++;
            }
        }
    }
}
```

**Explanation of Changes:**

1. **Array Slicing:**
    - In the mergeSort method, replaced the incorrect array slicing (array+1, array-1) with proper array-splitting methods (leftHalf and rightHalf).
2. **Fixed merging logic:**

- Removed the invalid increment and decrement operations (left++, right--) when passing arrays. The arrays are passed as they are.

3. **Merge logic:**
    - The merge method is updated to correctly merge the two sorted arrays (left and right) into the result array.

**[6] Multiply Matrices**

**Issues:**

1. **Incorrect Indexing in Multiplication:**
    - The indexing in the multiplication loop is wrong. The expressions first[c-1][c-k] and second[k-1][k-d] will lead to ArrayIndexOutOfBoundsException and incorrect results. The indices should be adjusted to access the correct elements from both matrices.
2. **Input for Second Matrix:**
    - The output mentions asking for the dimensions of the second matrix twice, which is misleading. The prompt should reflect the input correctly.
3. **Sum Calculation Logic:**
    - The logic for summing the products should be corrected to reset the sum for each element of the result matrix.

**Corrected Code:**

```
import java.util.Scanner;


class MatrixMultiplication {
  public static void main(String args[]) {
    int m, n, p, q;


    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of rows and columns of first
matrix:");
    m = in.nextInt();
    n = in.nextInt();

    int first[][] = new int[m][n];

    System.out.println("Enter the elements of first matrix:");
    for (int c = 0; c < m; c++)
      for (int d = 0; d < n; d++)
        first[c][d] = in.nextInt();

    System.out.println("Enter the number of rows and columns of second
matrix:");
    p = in.nextInt();
    q = in.nextInt();

    // Check if matrices can be multiplied
    if (n != p) {
      System.out.println("Matrices with entered orders can't be multiplied
with each other.");
    } else {
      int second[][] = new int[p][q];
      int multiply[][] = new int[m][q];

      System.out.println("Enter the elements of second matrix:");
      for (int c = 0; c < p; c++)
        for (int d = 0; d < q; d++)
```

```
                second[c][d] = in.nextInt();

            // Perform matrix multiplication
            for (int c = 0; c < m; c++) {
                for (int d = 0; d < q; d++) {
                    multiply[c][d] = 0; // Initialize the element to 0
                    for (int k = 0; k < n; k++) {
                        multiply[c][d] += first[c][k] * second[k][d]; // Correct indexing
                    }
                }
            }

            // Display the product of the matrices
            System.out.println("Product of entered matrices:");
            for (int c = 0; c < m; c++) {
                for (int d = 0; d < q; d++)
                    System.out.print(multiply[c][d] + "\t");

                System.out.print("\n");
            }
        }
        in.close();
    }
}
```

**Key Changes Made:**

1. **Corrected Indexing:**

- The multiplication logic now uses first[c][k] and second[k][d] to correctly multiply the elements of the matrices.

2. **Sum Initialization:**
   - Each element in the result matrix (multiply[c][d]) is initialized to 0 before accumulating the sum.
3. **Clarified Prompts:**
   - The prompts for entering matrix sizes and elements have been made clear and concise.

**[7] Quadratic Probing**

**Issues Identified:**

1. **Syntax Error in the Insertion Logic:**
   - The expression i + = (i + h / h--) % maxSize; has a syntax error. It should be i = (i + h * h) % maxSize;.
2. **Incorrect Logic in Hashing and Rehashing:**
   - The way to calculate the next index in the probing sequence is incorrect. It should involve incrementing the h variable for each probe attempt correctly.
3. **Incorrect Output and Input Handling:**
   - The printHashTable method should display the keys and values correctly. The input handling in the main method should prompt for key-value pairs in a loop.
4. **The get Method Logic:**
   - The increment logic in the get method also has an issue in the probing sequence.
5. **Cleanup of Unused Imports:**
   - Remove unnecessary imports to keep the code clean.

**Corrected Code:**

```
import java.util.Scanner;


/** Class QuadraticProbingHashTable **/
```

```
class QuadraticProbingHashTable {
   private int currentSize, maxSize;
   private String[] keys;
   private String[] vals;

   /** Constructor **/
   public QuadraticProbingHashTable(int capacity) {
      currentSize = 0;
      maxSize = capacity;
      keys = new String[maxSize];
      vals = new String[maxSize];
   }

   /** Function to clear hash table **/
   public void makeEmpty() {
      currentSize = 0;
      keys = new String[maxSize];
      vals = new String[maxSize];
   }

   /** Function to get size of hash table **/
   public int getSize() {
      return currentSize;
   }

   /** Function to check if hash table is full **/
   public boolean isFull() {
      return currentSize == maxSize;
   }

   /** Function to check if hash table is empty **/
   public boolean isEmpty() {
      return getSize() == 0;
   }

   /** Function to check if hash table contains a key **/
   public boolean contains(String key) {
      return get(key) != null;
   }

   /** Function to get hash code of a given key **/
   private int hash(String key) {
      return (key.hashCode() % maxSize + maxSize) % maxSize; // Ensuring
non-negative
   }

   /** Function to insert key-value pair **/
   public void insert(String key, String val) {
      if (isFull()) {
          System.out.println("Hash Table is full, cannot insert.");
          return;
      }

      int tmp = hash(key);
      int i = tmp, h = 1;

      do {
          if (keys[i] == null) {
              keys[i] = key;
              vals[i] = val;
              currentSize++;
              return;
          }
          if (keys[i].equals(key)) {
              vals[i] = val;
              return;
          }
          i = (tmp + h * h) % maxSize; // Corrected probing
          h++;
      } while (i != tmp);
   }

   /** Function to get value for a given key **/
   public String get(String key) {
      int i = hash(key), h = 1;

      while (keys[i] != null) {
          if (keys[i].equals(key)) {
              return vals[i];
          }
          i = (i + h * h) % maxSize; // Corrected probing
```

```
            h++;
        }
        return null;
    }

    /** Function to remove key and its value **/
    public void remove(String key) {
        if (!contains(key)) {
            return;
        }

        /** Find position key and delete **/
        int i = hash(key), h = 1;

        while (!key.equals(keys[i])) {
            i = (i + h * h) % maxSize;
            h++;
        }

        keys[i] = vals[i] = null;

        /** Rehash all keys **/
        for (i = (i + h * h) % maxSize; keys[i] != null; i = (i + h * h) % maxSize) {
            String tmp1 = keys[i], tmp2 = vals[i];
            keys[i] = vals[i] = null;
            currentSize--;
            insert(tmp1, tmp2);
        }
        currentSize--;
    }

    /** Function to print HashTable **/
    public void printHashTable() {
        System.out.println("\nHash Table: ");
        for (int i = 0; i < maxSize; i++)
            if (keys[i] != null)
                System.out.println(keys[i] + " " + vals[i]);
        System.out.println();
    }
}

/** Class QuadraticProbingHashTableTest **/
public class QuadraticProbingHashTableTest {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Hash Table Test\n\n");
        System.out.print("Enter size: ");

        /** Make object of QuadraticProbingHashTable **/
        QuadraticProbingHashTable qpht = new
QuadraticProbingHashTable(scan.nextInt());

        char ch;
        /** Perform QuadraticProbingHashTable operations **/
```

```
do {
  System.out.println("\nHash Table Operations\n");
  System.out.println("1. Insert");
  System.out.println("2. Remove");
  System.out.println("3. Get");
  System.out.println("4. Clear");
  System.out.println("5. Size");

  int choice = scan.nextInt();
  switch (choice) {
    case 1:
      System.out.println("Enter key and value");
      qpht.insert(scan.next(), scan.next());
      break;
    case 2:
      System.out.println("Enter key");
      qpht.remove(scan.next());
      break;
    case 3:
      System.out.println("Enter key");
      System.out.println("Value = " + qpht.get(scan.next()));
      break;
    case 4:
      qpht.makeEmpty();
      System.out.println("Hash Table Cleared\n");
      break;
    case 5:
```

```
                System.out.println("Size = " + qpht.getSize());
                break;
            default:
                System.out.println("Wrong Entry \n ");
                break;
        }

        /** Display hash table **/
        qpht.printHashTable();

        System.out.println("\nDo you want to continue (Type y or n) \n");
        ch = scan.next().charAt(0);
    } while (ch == 'Y' || ch == 'y');

    scan.close();
  }
}
```

**Key Changes Made:**

1. **Fixed Syntax Errors:**
   - Corrected the syntax for the index update in the insert, get, and remove methods.
2. **Handling Negative Hash Codes:**
   - Modified the hash method to ensure the returned index is non-negative by adjusting with maxSize.
3. **Enhanced Input Validation:**
   - Added checks for whether the hash table is full before inserting a new key-value pair.
4. **Code Formatting:**

- Cleaned up the code for better readability and consistency.

**[8] Sorting Array**

**Issues Identified:**

1. **Class Naming Error:**
   - The class name Ascending _Order contains a space, which is invalid in Java. Class names should not have spaces.
2. **Sorting Logic Error:**
   - The condition in the outer loop for (int i = 0; i >= n; i++); has the wrong comparison operator (>= instead of <), and there is an unnecessary semicolon at the end of the loop.
3. **Incorrect Sorting Condition:**
   - The sorting condition inside the nested loop should use if (a[i] > a[j]) to perform a swap for ascending order.
4. **Output Formatting:**
   - The output logic in the printing loop can be simplified for better readability.

**Corrected Code:**

```
import java.util.Scanner;


public class AscendingOrder {
    public static void main(String[] args) {
        int n, temp;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter no. of elements you want in array: ");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter all the elements:");


        // Read elements into the array
        for (int i = 0; i < n; i++) {
            a[i] = s.nextInt();
        }

        // Sorting the array in ascending order using bubble sort
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (a[j] > a[j + 1]) {
                    // Swap a[j] and a[j + 1]
                    temp = a[j];
                    a[j] = a[j + 1];
                    a[j + 1] = temp;
                }
            }
        }

        // Output the sorted array
        System.out.print("Ascending Order: ");
        for (int i = 0; i < n; i++) {
            System.out.print(a[i]);
            if (i < n - 1) {
                System.out.print(", ");
            }
        }
        System.out.println(); // New line for better output formatting
    }
}
```

**Key Changes Made:**

1. **Class Name Correction:**
   - Renamed the class to AscendingOrder without spaces.
2. **Fixed Loop Conditions:**
   - Changed the outer loop condition to i < n - 1.
3. **Corrected Sorting Logic:**
   - Modified the condition for swapping elements to sort in ascending order.
4. **Improved Output Formatting:**
   - Simplified the print statement for outputting the sorted array and added a new line at the end.

**[9] Stack Implementation**

**Issues Identified:**

1. **Push Method Error:**
   - The push method is decrementing top before assigning the value to the stack, which results in an incorrect index and causes array index out-of-bounds issues.
2. **Pop Method Error:**
   - The pop method is incrementing top, but it should decrement it instead to reflect the removal of the top element.
3. **Display Method Logic Error:**
   - The condition in the for loop of the display method is incorrect (i > top should be i <= top), which means it won't display any elements.
4. **Display of Stack Elements:**
   - It's better to display the elements starting from the top of the stack to the bottom.

**Corrected Code:**

```
// Stack implementation in Java
```

```
import java.util.Arrays;

public class StackMethods {
    private int top;
    int size;
    int[] stack;

    public StackMethods(int arraySize) {
        size = arraySize;
        stack = new int[size];
        top = -1;
    }

    public void push(int value) {
        if (top == size - 1) {
            System.out.println("Stack is full, can't push a value");
        } else {
            top++; // Increment top before pushing the value
            stack[top] = value; // Assign the value to the current top position
        }
    }

    public void pop() {
        if (!isEmpty()) {
            System.out.println("Popped: " + stack[top]); // Display the popped
value
            top--; // Decrement top to remove the value
        } else {
            System.out.println("Can't pop...stack is empty");
        }
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public void display() {
        if (isEmpty()) {
            System.out.println("Stack is empty");
            return;
        }
        System.out.print("Stack elements: ");
        for (int i = top; i >= 0; i--) { // Display from top to bottom
            System.out.print(stack[i] + " ");
        }
        System.out.println();
    }
}

public class StackReviseDemo {
    public static void main(String[] args) {
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);

        newStack.display(); // Display stack elements
        newStack.pop(); // Pop elements and display
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display(); // Display remaining stack elements
    }
}
```

**Key Changes Made:**

1. **Fixed Push Logic:**
    - Changed the push method to increment top before assigning the value to the stack.
2. **Corrected Pop Logic:**
    - Modified the pop method to decrement top after displaying the popped value.
3. **Updated Display Logic:**
    - Adjusted the display method to iterate from top down to 0 to show the stack elements correctly.
4. **Output when Popping:**
    - Added a print statement in the pop method to indicate which value is being popped.

**[10] Tower of Hanoi**

**Issues Identified:**

1. **Increment and Decrement Operators:**

- You used ++ and -- incorrectly in the recursive call. The syntax topN ++ and inter-- doesn’t change the variables for the subsequent calls. You need to pass the variables as-is, not modify them during the call.

2. **Character Manipulation:**
   - You should not perform arithmetic operations on characters (like from + 1 or to + 1). Instead, you should pass the characters directly without modifications.
3. **Base Case Output:**
   - The output statement for the base case is correct, but you might want to improve clarity or include disk numbers dynamically.

**Corrected Code:**

```
// Tower of Hanoi
public class MainClass {
  public static void main(String[] args) {
    int nDisks = 3;
    doTowers(nDisks, 'A', 'B', 'C');
  }


  public static void doTowers(int topN, char from, char to, char inter) {
    if (topN == 1) {
      System.out.println("Disk 1 from " + from + " to " + to);
    } else {
      doTowers(topN - 1, from, inter, to); // Move topN-1 disks from 'from' to 'inter'
      System.out.println("Disk " + topN + " from " + from + " to " + to); // Move the nth disk to 'to'
      doTowers(topN - 1, inter, to, from); // Move the disks from 'inter' to 'to'
    }
  }
}
```

**Key Changes Made:**

1. **Recursive Call Fixes:**
    - Updated the recursive calls to doTowers(topN - 1, ...) instead of using increment/decrement operators.
2. **Character Parameters:**
    - Passed character parameters as-is without arithmetic operations.