Task1:

In the attached file **FileList.txt** you will find a list of files that have been changed during the installation of the VLC video player.

You are given the following class:

public class FsEntry

{

// name of a directory or a file

public string Name {get; set;}

// full path to the directory of file with a trailing slash

public string Path {get; set;}

// return type of the entry depending on the Path and Children

public bool IsDirectory => ...

public bool IsFile => ...

public bool IsShortcut => ...

// subdirectories or files in directory

public List<FsEntry> Children { get; set; }

}

Your task is to parse the list of paths into a hierarchical tree using LINQ.

Each of these elements must have their own subelements in Children, depending on their subfolders.

The result should be a method that return List<FsEntry>

**Solution:**

var file = File.ReadAllLines("FileList.txt");

var tree = FsEntry.BuildFileTree(file);

FsEntry.PrintTree(tree);

Console.ReadKey();

public class FsEntry

{

// name of a directory or a file

public string Name { get; set; }

// full path to the directory of file with a trailing slash

public string Path { get; set; }

// return type of the entry depending on the Path and Children

public bool IsDirectory => Children.Count > 0;

public bool IsFile => !IsDirectory && !IsFile;

public bool IsShortcut => Name.EndsWith(".lnk");

// subdirectories or files in directory

public List<FsEntry> Children { get; set; }

public static List<FsEntry> BuildFileTree(IEnumerable<string> paths)

{

var pathParts = paths

.Select(path => path.Split('\\'))

.ToList();

return BuildTree(pathParts, "");

}

private static List<FsEntry> BuildTree(List<string[]> partsList,

string basePath)

{

return partsList.Where(p => p.Length > 0)

.GroupBy(p => p[0])

.Select(group =>

{

var name = group.Key;

var fullPath = basePath + name + "\\";

var childrenParts = group.Where(g => g.Length > 1)

.Select(g => g.Skip(1).ToArray()).ToList();

var children = BuildTree(childrenParts, fullPath);

return new FsEntry

{

Name = name,

Children = children,

Path = fullPath

};

}).ToList();

}

public static void PrintTree(List<FsEntry> entries, int indent = 0)

{

foreach (var entry in entries)

{

Console.WriteLine($"{new string(' ', indent)}- {entry.Name} " +

$"{(entry.IsDirectory ? "[DIR]" : entry.IsShortcut ? "[LNK]" : "[FILE]")}");

if (entry.Children != null && entry.Children.Count > 0)

PrintTree(entry.Children, indent + 2);

}

}

}

In this solution we define if entry is directory by checking if it has any child nodes, if so it is a directory, otherwise it is not considered as directory.

A file is defined if entry is neither the directory nor the shortcut.

While checking for entry to be a shortcut we use C# string method “EndsWith” and check if it meets the requirement by ending like ‘.lnk’.

Method BuildTree accepts as parameters a list of arrays string and a base path that will be used later to create the full path.

Moving on to the creating a method that would build the file tree this method split every line into array and then turn them into a list of string arrays that we then pass to the method called “BuildTree” that is returned by this method. Then we filter any arrays that are empty, group the remaining array by their first element that represents the current level. For each group first element is considered as name of the current node, full path is created by appending name to the path and backslash. We skip the first part to collect the remaining segments and treat them as child parts. This method is called recursively to process child parts. Then we create a new object “FsEntry” for each group and assign its propertioies accordingly to defined ones.

The PrintTree method was created verify and display the result.

Task2:  
In 5–8 sentences, describe in your own words what the full lifecycle of a web application in the cloud looks like.

Where does it start? What are the key stages (e.g., development, testing, deployment, monitoring)?  
Try to imagine it as if you were working on a real project

Answer:

Cloud application lifecycle starts with defining the purpose of application, what it will be required to do, how it will be used, what type it will be. On this stage it is important to define the security of the app, how it will be maintained and the network that is crucial for data transfer.

Then we define tools, frameworks and cloud services to build application. Next is the developing stage. This stage is split into steps by which the application will be made. Developers use Git to ensure the current version is functioning correctly before moving on to the next one. Each implementation of features is followed with Unit Test to minimize the possible bugs on testing stage.

During the testing stage team test application for the possible malfunctions using Unit Tests, manual testing.

Then application moved to the deployment stage where team verifies that application works correctly.

Then comes the lifecycle management, with the requirements of the client the application is being maintained, being implemented with new features.

Task3:  
Below is a simple Dockerfile:

FROM python:3.12

COPY . /app

WORKDIR /app

RUN pip install -r requirements.txt

CMD ["python", "main.py"]

Write a brief explanation (1–2 sentences per line) of what each line does.

Answer:  
On the first line we use the base image that contains Python version 3.12.

On the second line we copy all the files from the current directory into the application folder.

On the third line we set working directory to the app folder.

On the fourth line we install all Python dependencies listed in requirements.txt.

Finally we set the default command to run the app.

Task4:  
In your own words, explain the purpose of DevOps. How would you describe it to someone with no background in IT?

Answer:

DevOps is an approach to software development that unites developers and operators so they would work as a team. It is used to speed up applications development, automate its deployment and ensure that everything meets the required standards.

You can compare it to a kitchen where developers are chefs and operators are waiters. They work together to ensure that every dish is made on time and they would be delivered to the client`s table without any difficulties.