In order to run my work follow these instructions:

1. **Task 1:**
    a. Run the task1.exe file
2. **Task 2 and Task 3:**
    a. Open myserver folder and run myServer.exe file
    b. Run Client.html or press here
    c. Press LET'S START button.

Documentation:

Task1: task1.c

Header usage:

a. Stdio.h
b. String.h
c. Windows.h
d. Subauth.h-
e. Winternl.h – holds all PEB structures that needed to the mission

In this mission we can separate between two parts. First part is working with peb structures and extract the dll's names, and second part is save those names in some data structure.

let as review the first part:

Structure:

```
typedef NTSTATUS (NTAPI *NTQUERYINFORMATIONPROCESS)(
    IN HANDLE ProcessHandle,
    IN PROCESSINFOCLASS ProcessInformationClass,
    OUT PVOID ProcessInformation,
    IN ULONG ProcessInformationLength,
    OUT PULONG ReturnLength OPTIONAL
    );
```

In order to obtain current process PEB structure I had to use NtQueryInformationProcess function and to do so I had to use NTSTATUS as it explained here.

Functions:

```
PROCESS_BASIC_INFORMATION* currPbiProcess(){…}
```

The function currPbiProcess() does not get any element and returns pointer to Process_Basic_Information structure which holds the pointer to the PEB structure I'm looking for. That function organize all the element necessary to the NtQueryInformationProcess function and then calls her.

```
dllLink * createDllNamesList( PPEB peb){…}
```

The function createDllNamesList(PPEB peb) gets pointer to PEB structure and returns a link-list which holds all dll's names. In this function we extract the PEB_LDR_DATA structure (explained here) from the PEB and then copy all the dll's name into the list.

Now let as review the second part:

Structure:

```
typedef struct _link{     //Circular linked list
  UNICODE_STRING * name;
  struct _link * next;
  struct _link * prev;
}dllLink;
```

I choose to use a circular linked list in order to save all dll's names. Each link in the list holds a dll file name, a pointer to the next link in the list and a pointer to the previous link in the list.

Function:

```
dllLink * addlast(dllLink* head,UNICODE_STRING * name){…}
```

The addlast function gets a pointer to the head of a list and a dll's file name. The function create a new link and adds it to the list. the function returns a pointer to the list.

```
void printList( dllLink * head, char flag){…}
```

the printList function gets a pointer to the head of a list and a flag which indicate where to print the dll's names, if flag=0 the function will print her output to stdout else it will print to a file called dllfiles.txt. The function doesn't return a value but it print the name element for each link in the list.

Task 2: myServer.py

Imported libraries:

```
import SimpleHTTPServer
import SocketServer
import time
from os import curdir, sep, system
from io import BytesIO
from myDataHandler import data_handler  //created by me
```

The main work in that mission was to create a http handler which handle the requests properly. I created MyServer class which hold these next function:

```
    def do_POST(self):
```

the do_POST() function which handle any post request received from the client.

(that kind of requests are used while working with the dll's information table)

```
    def do_GET(self):
```

the do_GET() function which handle any get request received from the client.

(that kind of requests are used mainly to load the html pages properly)

Task 2 : MyDataHandler.py

Imported libraries:

```
import csv
import codecs
from os import curdir, sep, system
import subprocess
import time
```

this file holds all the work with the csv file and holds the data structure which hold the csv file information. In this file I created a class called data_handler. The data_handler class holds these functions:

```
    def __init__(self):
```

this function set all the information needed to the class. It read the csv file and organize all the information in the filed self.MyData which hold a dictionary orderd by dll's names.

```
    def printdatabase(self):
```

This function has been made only for checking and debugging purpose, it print self.MyData information.

```
    def createTable(self, data):
```

This function create mytable.html with the information that received by the data element

If mytable.html is exist the function over write it.

```
    def search(self, s_key):
```

This function is related to the search button, it get a key to search in the self.MyData structure and create a relevant data which been send to the createTable function

```
    def run_task1(self):
```

This function is running in a sub process the task1 file with an element so that the output will by printed to dllfiles.txt

```
    def get_task1_names(self):
```

This function is extract all dll's name from the dllfiles.txt file and return a list which holds them.

```
    def loadtask1(self):
```

This function is related to Load Task1 Dll's button. It call's run_task1 and get_task1_names function and creates a relevant data which send to the createTable function

```
    def clear(self):
```

This function create an empty string data which send to the createTable function

```
    def loadSelected(self,selected):
```

This function is related to Show selected button and it get all the selected entries from the table then it loop over mytable.html file and over-write only the selected elements.

Task 3: client

I have choose to create a website in order to implement this mission in order to do so I have download most of the html,js and css code from here and modify the files to my own needs .