מערכות הפעלה – פתרון תרגיל 1

```
#include "stdafx.h"
#include <windows.h>
#include <assert.h>
#include <conio.h>

#define UNICODE_SIGNATURE WORD(0xFEFF)

DWORD getClusterSize(LPCWSTR folder_path, LPDWORD clusterSize)
{
	TCHAR drive[MAX_PATH + 1];
	DWORD diskPathLen;
	TCHAR fileName[MAX_PATH];
	BOOL bRes;

	/* get the drive */
	_tsplitpath(folder_path, drive, NULL, fileName, NULL);

	/* add / at the end */
	diskPathLen = _tcslen(drive);
	drive[diskPathLen] = '\\';
	drive[diskPathLen + 1] = '\0';

	/* Get the size of a cluster */
	DWORD
lpSectorsPerCluster=0,lpBytesPerSector=0,lpNumberOfFreeClusters=0,lpTotalNum
berOfClusters=0;
	bRes =
GetDiskFreeSpace(drive,&lpSectorsPerCluster,&lpBytesPerSector,&lpNumberOfFre
eClusters,&lpTotalNumberOfClusters);
	if (bRes == 0) { return FALSE; }

	/* Calc the size on the disk */
	*clusterSize = lpSectorsPerCluster*lpBytesPerSector;

	return TRUE;
}

/*	computes the size of the file on the disk and returns the result
	in diskFileSize. On error returns 0;*/
void calcSizeOnDisk(const DWORD clusterSize, const DWORD actualSize, LPDWORD
diskFileSize)
{
	/* Calc the size on the disk */
	DWORD clustersUsed = (actualSize/clusterSize);

	/* unless the file size takes complete clusters */
	if (actualSize % clusterSize != 0)
	{
		/* round up */
		clustersUsed++;
	}

	*diskFileSize = clustersUsed*clusterSize;
}

DWORD calcHash(LPCWSTR file_path, DWORD fileSize, LPDWORD hash)
{
	BOOL bRes;

	/* open the file */
	HANDLE
hFile=CreateFile(file_path,GENERIC_READ,FILE_SHARE_READ,NULL,OPEN_EXISTING,F
ILE_FLAG_RANDOM_ACCESS,NULL);
	assert(hFile!=INVALID_HANDLE_VALUE);
```

```
	/* read the first word */
	BYTE byteFirst=0,byteLast=0;
	DWORD dwRead=0;

	/*if this is not an empty file */
	if (fileSize == 0)
	{
		/* return 0 as hash */
		*hash = 0;
	}
	else
	{
		/* read the first byte */
		bRes=ReadFile(hFile,&byteFirst,sizeof(byteFirst),&dwRead,NULL);
		assert(bRes && (sizeof(byteFirst)==dwRead));

		/* read the last byte */
		SetFilePointer(hFile, fileSize-1, NULL, FILE_BEGIN);
		bRes=ReadFile(hFile,&byteLast,sizeof(byteLast),&dwRead,NULL);
		assert(bRes && (sizeof(byteLast)==dwRead));

		/* calc the hash */
		*hash = (byteFirst + byteLast)/2;
	}

	/* close the file */
	CloseHandle(hFile);

	return TRUE;
}

//print format (unicode or ascii) for text files
int _tmain(int argc, _TCHAR* argv[])
{
	//first is executable path, second is path to target folder
	//and the third is the output path
	assert(argc==3);
	LPCTSTR srcFolder=argv[1];
	LPCTSTR outFolder=argv[2];
	DWORD clusterSize = 0;
	BOOL bRes;
	TCHAR output[2*MAX_PATH];

	/* get the drive clusterSize */
	bRes = getClusterSize(srcFolder, &clusterSize);
	assert(bRes && (clusterSize > 0));

	/* get the output file path */
	TCHAR full_out_file_path[MAX_PATH];
	_tcscpy(full_out_file_path,outFolder);

	/* open a file to be written as unicode*/
	HANDLE
hOutFile=CreateFile(full_out_file_path,GENERIC_WRITE,0,NULL,CREATE_ALWAYS,FI
LE_FLAG_SEQUENTIAL_SCAN,NULL);
	assert(hOutFile!=INVALID_HANDLE_VALUE);
	DWORD uniSig = UNICODE_SIGNATURE;
	DWORD bytesWritten=0;
	bRes=WriteFile(hOutFile,&uniSig,2,&bytesWritten,NULL);
	assert(bRes && bytesWritten == 2);

	/* create the search string */
	TCHAR search_string[MAX_PATH];
	_tcscpy_s(search_string,srcFolder);
	_tcscat_s(search_string,_T("*"));

	/* Get the iterator*/
```

```
	WIN32_FIND_DATA w32fd;
	ZeroMemory(&w32fd,sizeof(w32fd));
	HANDLE hFindFile=FindFirstFile(search_string,&w32fd);

	/* make sure there where no errors*/
	bRes=(hFindFile!=INVALID_HANDLE_VALUE);

	while(bRes)
	{
		//Make sure that this is not a folder, a "."/".." record or a
system file
		if (((w32fd.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM) == 0) &&
((w32fd.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) == 0))
		{
			/* calc the file size (file size is limited to 2MB) */
			DWORD fileSize = (w32fd.nFileSizeHigh * (MAXDWORD+1)) +
w32fd.nFileSizeLow;

			/* calc the file size on the disk */
			DWORD diskFileSize = 0;
			calcSizeOnDisk(clusterSize, fileSize, &diskFileSize);

			/* calc the file hash */
			DWORD fileHash = 0;
			TCHAR full_file_path[MAX_PATH];
			_tcscpy(full_file_path,srcFolder);
			_tcscat(full_file_path,w32fd.cFileName);
			bRes=calcHash(full_file_path,fileSize,&fileHash);
			assert(bRes);

			/* print to file */

	bRes=_stprintf(output,_T("%s\t%d\t%d\t%d\r\n"),w32fd.cFileName,fileSiz
e,diskFileSize,fileHash);
			assert(bRes != 0);

	bRes=WriteFile(hOutFile,&output,_tcslen(output)*sizeof(_TCHAR),&bytesW
ritten,NULL);
			assert(bRes != 0 &&
_tcslen(output)*sizeof(_TCHAR)==bytesWritten);
		}

		bRes=FindNextFile(hFindFile,&w32fd);
	}

	/* check for error */
	assert(GetLastError()==ERROR_NO_MORE_FILES);

	/* close the file finder */
	FindClose(hFindFile);
	CloseHandle(hOutFile);

	return 0;

}
```