Name : Muhammad Ishraf Shafiq Zainuddin

ID : 200342741

Assgn : 1

Part 1

//main.cpp

```
#include "parse.h"
#include <iostream>

using namespace std;

int main()
{
   string tokens [MAX_COMMAND_LINE_ARGUMENTS], commandLine;
   int tokenCount, i;
   char ** wordsC;
   bool cleanC, commandEntered;

   cout << "Welcome humans!!" << endl << endl;
   cout << "To Tacos or not To Tacos, that's the question... " << endl << endl;

   //Material from the Lab (HALmod.cpp - GetCommand Function)
   do
   {
      cout << "User input: "; //Asking user for input
      while (1)
      {
         getline (cin, commandLine);          //Get user input
         commandEntered = CheckForCommand ();
         if (commandEntered)
         {
            break;
         }
```

```
        }
    } while (commandLine.length () == 0);

    tokenCount = TokenizeCommandLine (tokens, commandLine);  //Calling TokenizeCommandLine
Func.

    return tokenCount;

    } while (1);

    CleanUpCArray (wordsC, tokenCount); //CleanUpCArray
    Print (wordsC, tokenCount); //Prints the results from the parser

    return 0;
}
```

//parse.cpp

```
#include "parse.h"

//Materials from the Lab (HALmod.cpp)
/*int GetCommand (string tokens [])
{
    string commandLine;
    bool commandEntered;
    int tokenCount;

    do
    {
        //The below line is in Dr. Hilderman's code, we won't need it for the lab
        //BlockSignals ("HALshell");
        cout << "HALshell> ";
        while (1)
        {
            getline (cin, commandLine);
```

```
            commandEntered = CheckForCommand ();
            if (commandEntered)
            {
               break;
            }

         }
         //The below line is in Dr. Hilderman's code, we won't need it for the lab
         //UnblockSignals ("HALshell");
      } while (commandLine.length () == 0);

      tokenCount = TokenizeCommandLine (tokens, commandLine);

      return tokenCount;
   }*/

   //Materails from the Lab (Halmod.cpp)
   int TokenizeCommandLine (string tokens [], string commandLine)
   {
      char *token [MAX_COMMAND_LINE_ARGUMENTS];
      char *workCommandLine = new char [commandLine.length () + 1];
      int i;
      int tokenCount;

      for (i = 0; i < MAX_COMMAND_LINE_ARGUMENTS; i ++)
      {
         tokens [i] = "";
      }
      strcpy (workCommandLine, commandLine.c_str ());
      i = 0;
      if ((token [i] = strtok (workCommandLine, " ")) != NULL)
      {
         i ++;
         while ((token [i] = strtok (NULL, " ")) != NULL)
         {
```

```
            i ++;
        }
    }
    tokenCount = i;

    for (i = 0; i < tokenCount; i ++)
    {
        tokens [i] = token [i];
    }

    delete [] workCommandLine;

    return tokenCount;
}

//Materials from the Lab (HALmod.cpp) *Do not touch the below function
bool CheckForCommand ()
{
    if (cullProcess)
    {
        cullProcess = 0;
        cin.clear ();
        cout << "\b\b  \b\b";
        return false;
    }

    return true;
}

//Converting tokens into a c version of an array of words and returns the pointer to that array
char ** ConvertToC (string tokens [], int tokenCount)
{
  char ** words;
  words = (char**)malloc(sizeof(char*)* tokenCount);
  int i;
```

```
  for (i = 0; i < tokenCount; i++)
  {
    words[i] = strdup(tokens[i].c_str());
  }

  return words;
}

//This cycles through the c string version of the array and frees up any memory that has been allocated
bool CleanUpCArray (char ** cTokens, int tokenCount)
{
 int i;

 for (i = 0; i < tokenCount; i++)
 {
  free(cTokens[i]);
 }

}

//Cycles through the c string version of the words and prints each word
void Print (char ** cTokens, int tokenCount)
{
 int i;

 for (i = 0; i < tokenCount; i++)
 {
  cout << cTokens[i] << " ";
 }

 cout << endl << endl;
}
```

```
//Materials from the Lab (HALmod.cpp) *A very paired down version of Dr. Hilderman's function
void ProcessCommand (string tokens [], int tokenCount)
{
   if (tokens [0] == "shutdown" || tokens [0] == "restart" || tokens[0] == "lo")
   {
      ShutdownAndRestart (tokens, tokenCount);
      // if no error, then never returns
      return;
   }
   else{
     // this is where the Print function should be called in
     char ** cTokens;
     cTokens = ConvertToC(tokens, tokenCount);
     Print(cTokens, tokenCount);

   }
}

//Materials from the Lab (HALmod.cpp)
void ShutdownAndRestart (string tokens [], int tokenCount)
{
   if (tokenCount > 1)
   {
      cout << "HALshell: " << tokens [0] << " does not require any arguments" << endl;
      return;
   }

   cout << endl;
   cout << "PARshell: terminating ..." << endl;
   //The below lines are in Dr. Hilderman's code, we won't need it for the lab
   //system ("HALshellCleanup");
   //usleep (SLEEP_DELAY);
   //SendCommandLineToHALos (tokens, tokenCount);
   exit (0);
}
```

//parse.h

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <cstring>


using namespace std;

//The following two lines come from HALglobals.h
const int MAX_COMMAND_LINE_ARGUMENTS = 8;
const int SLEEP_DELAY = 100000;

//The following lines are materials from the Lab (HALmod.h)
//int GetCommand (string tokens []);
int TokenizeCommandLine (string tokens [], string commandLine); //Tokenize string from main.cpp
bool CheckForCommand ();
void ProcessCommand (string tokens [], int tokenCount);
void ShutdownAndRestart (string tokens [], int tokenCount);
static volatile sig_atomic_t cullProcess = 0;

//Converting tokens into a c version of an array of words and returns the pointer to that array
char ** ConvertToC (string tokens [], int tokenCount);

//This cycles through the c string version of the array and frees up any memory that has been allocated
bool CleanUpCArray (char ** cTokens, int tokenCount);

//Cycles through the c string version of the words and prints each word
void Print (char ** cTokens, int tokenCount);
```

Part 2

- g++ main.cpp -o main

- ls -l

- ./main

Part 3

(a) – tic.c file, line 2015. By asking user the input by (Commenting) and it's is literally a 'parsecode' function.

(b) – console.c file, line 2400, "commandDone(console);"

(c) – Yes, Tic-80 does call external commands but cannot access the host file system