Name : Muhammad Ishraf Shafiq Zainuddin

ID : 200342741

Lab Assignment : 1

(1) //HALmod.cpp

```
#include "HALmod.h"
```

char ** ConvertToC (string tokens [], int tokenCount)

```
{
   char ** words;
   words = (char**)malloc(sizeof(char*)* tokenCount);
   for (int i = 0; i < tokenCount; i++)
   {
     words[i] = strdup(tokens[i].c_str());
   }
    return words;
}
```

bool CleanUpCArray (char ** cTokens, int tokenCount)

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

void PrintReverse (char ** cTokens, int tokenCount)

```
{
  for (int i = tokenCount; i > 0; i--)
  {
    cout << cTokens[i] << " ";
  }
  cout << "\n\n";
}
```

```
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;
}

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

```
    return true;
}

//This is 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 PrintReverse function should be called in
      char ** cTokens;
      cTokens = ConvertToC(tokens, tokenCount);
      PrintReverse(cTokens, tokenCount);
    }
}

void ShutdownAndRestart (string tokens [], int tokenCount)
{
    if (tokenCount > 1)
    {
        cout << "HALshell: " << tokens [0] << " does not require any arguments" << endl;
        return;
    }
    cout << endl;
    cout << "HALshell: 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);
}
```

(2) //HALmod.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;

int GetCommand (string tokens []);
int TokenizeCommandLine (string tokens [], string commandLine);
bool CheckForCommand ();
void ProcessCommand (string tokens [], int tokenCount);
void ShutdownAndRestart (string tokens [], int tokenCount);
// extend functions from the lab
char ** ConvertToC (string tokens [], int tokenCount);
bool CleanUpCArray (char ** cTokens, int tokenCount);
void PrintReverse (char ** cTokens, int tokenCount);
static volatile sig_atomic_t cullProcess = 0;
```

(3.1) //main.cpp (Part 1)

1. (a) The space between words, “ “.

(b) Because strings are much easier to use than c strings especially regarding dynamic memory allocation

2. 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 PrintReverse function should be called in
char ** cTokens;
cTokens = ConvertToC(tokens, tokenCount);
PrintReverse(cTokens, tokenCount);
}
}
```

(3.2) //main.cpp (Part 2)

```
#include "HALmod.h"

int main()
{
   string tokens [MAX_COMMAND_LINE_ARGUMENTS];
   int tokenCount;
   char **words;

   do
   {
      ConvertToC (tokens[], tokenCount);
      tokenCount = GetCommand (tokens);
      ProcessCommand (tokens, tokenCount);
      CleanUpCArray (words, tokenCount);
      PrintReverse (words, tokenCount);
   } while (1);

   return 0;
}
```

(4) Script of Run

- gcc main.cpp -o main

- ./main