CS3026 Assessment 1 Memory Management Report

Name: Hasan Ahmedov

ID: 51768277

1. Description

This program represents an implementation of a simple memory management and allocation system.

The application operates with our own simplified version of the function malloc() that allows us to allocate memory segments, our version of function free() to deallocate these segments, and a function defrag() that defragments our "memory".

Our "memory" is a large array of bytes of length MAXMEM. Memory allocation is represented by managing a memory segmentation table that records which segments of memory are allocated and which parts are free. This segmentation table is a list of segment descriptors.

Initially the whole "memory" is one single free segment. An allocated segment is characterized by the pointer variable, for which the memory is allocated. The segmentation table is implemented as a linked list allowing entries to be dynamically added and removed.

The complete public interface for this memory management system consists of the following functions:

void initialize()
- this function initialises the segmentation table and the memory array

void * mymalloc (size_t size):
- this function "allocates" memory of a particular size and returns a pointer to the first byte of the allocated segment

void myfree (void * ptr):
- frees a previously allocated memory

void mydefrag(void ** ptrlist):
- defragments the memory by deleting unallocated segments and adding their size to the free part of memory

The helper functions used are **printmemory()**, **printsegmentdescriptor()**, **printsegmenttable()**, **isPrintable()**, **findFree()**, **insertAfter()**, **findSegment()** and **findFree()**. (further described with comments in **mymemory.c**)

2. How to run the program

(!) Before you start make sure u have all the files required in the same directory: **mymemory.c**, **mymemory.h**, **shell.c** and **Makefile**.

On Windows:

1) You will need to install C compiler if you don't have one, such as GCC (https://gcc.gnu.org/). If you are on a Windows computer, you can run the command '**gcc -v'** to check if it's already installed.

2) Open the command prompt by going to the **Start button** and typing '**cmd'** in the search or run bar (or click on Command Prompt if provided).

3) Change your directory to where you have your C program (in our case mymemory.c, mymemory.h, shell.c and the Makefile) using the command '**cd**'. We need to pass the name of the directory in which the program is stored.

    - **Example:** cd Desktop (if the program is already in the user directory)

4) Compile the source code by typing '**make**' in the Command Prompt (compilation is automatically done by the Makefile). (additional command '**make clean**' to clean up the compilation files)

5) Run the executable file by typing the name of the executable file without the extension (in our case '**shell**') and hit '**Enter**'.

On Linux:

1) Open the terminal window.

2) If you don't have a compiler already installed you will need to run the following apt-get commands in the terminal to install GNU c/c++ compiler:

```
$ sudo apt-get update
$ sudo apt-get install build-essential manpages-dev
```

3) Navigate to the program directory using the command '**cd**' and type in '**make**' to compile the program (since we have a Makefile). (additional command '**make clean**' to clean up the compilation files)

4) To run the executable just type in '**./<name of file>**' ('**./shell**' in our case) and hit '**Enter**'.

In case **Makefile** is missing you can compile the program manually using the following commands:

```
$ gcc -c mymemory.c
$ gcc -c shell.c
$ gcc -o shell shell.o mymemory.o
```

3. **Sample output**

Figure 1. Initialization and allocating first 3 memory segments

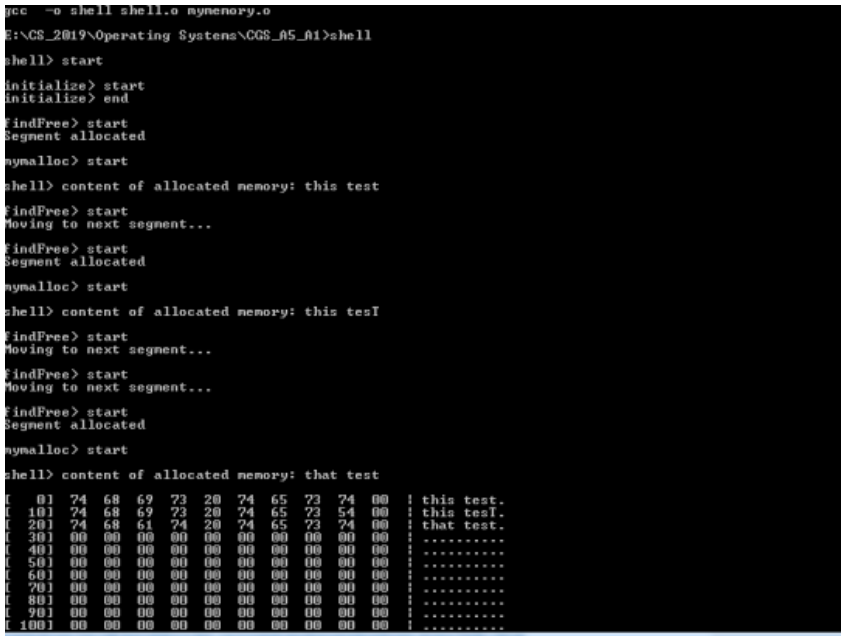

Figure 2. Further allocation and calling mydefrag()