Group 4

Jacob Thomas

Jake Batson

Laura Gibbs

Norman Benson

Rochak Kunwar

CS 470 Problem Set 07

# Exercises

1. In your own words, define each of the following terms:
   1. Stack division of memory

This is the location of memory used to store the local variables, so that they be accessed when a function is called and returned.

* 1. Heap division of memory

This is the location of memory where variables and objects are created when a program is running. The size of the program and vary depending on the programmer who controls what is created and deleted.

* 1. Code division of memory

The location of memory for compiled code such as functions, global variables, and constants.

* 1. Frame pointer

The current location in memory minus the size of the current function to allow the program to return to its position before the function call.

* 1. Stack pointer

This points to the next available position in memory on the stack. This act similar to a linked list and keeps a record of the last item put on the stack.

1. Describe in your own words how is allocation and freeing implemented in heap functions such as malloc/new and free/delete

When using malloc/new, memory is allocated in any random order within a block of memory meaning it must iterate through the block of memory to find false slot. Once a false slot is found it will mark it busy by setting it to true then returning the block of memories position.

When freeing the memory, it searches for the block of memory by its position and marking the block as freed.

# Problems

## Please start from the following representation of the call stack and the heap. Note that you will not need to draw the heap until #3.4. If a pointer exists in the call stack or the heap, please indicate where it points with an arrow. If a value exists in the call stack or the heap, please place that value in the corresponding memory location.

Hint: Draw this in a sheet of paper and take a picture with your phone. Insert the picture here:

3.1. After two local variables are declared in main() :

int main()

{

int integer = 42;

bool boolean = true;

//////// Problem 3.1 ////////////////



3.2. After three local variables are declared in function1()

// address: 0x4010000

void function1()

{

char text[8] = "Hello";

int num1 = 10;

int num2 = 20;

//////// Problem 3.2 ////////////////



3.3. At the end of function3()

// address: 0x4030000

void function3(char \* pointer,

int value,

int & reference)

{

\*pointer = 'X';

value = 99;

reference = 9999;

//////// Problem 3.3 ////////////////



3.4. After three local variables are declared in function2()

// address: 0x4020000

void function2()

{

int \* p1 = NULL;

char \* p2 = NULL;

float \* p3 = NULL;

//////// Problem 3.4 ////////////////



3.5. After p1 is allocated in function2()

// address: 0x4020000

void function2()

{

int \* p1 = NULL;

char \* p2 = NULL;

float \* p3 = NULL;

p1 = new int[32];

//////// Problem 3.5 ////////////////



3.6. After p2 and p3 are allocated in function2()

// address: 0x4020000

void function2()

{

int \* p1 = NULL;

char \* p2 = NULL;

float \* p3 = NULL;

p1 = new int[32];

p2 = new char[60];

p3 = new float[8];

//////// Problem 3.6 ////////////////



3.7. After p2 is deleted in function2()

// address: 0x4020000

void function2()

{

int \* p1 = NULL;

char \* p2 = NULL;

float \* p3 = NULL;

p1 = new int[32];

p2 = new char[60];

p3 = new float[8];

delete [] p2;

//////// Problem 3.7 ////////////////



3.8. After p2 is allocated again in function2()

// address: 0x4020000

void function2()

{

int \* p1 = NULL;

char \* p2 = NULL;

float \* p3 = NULL;

p1 = new int[32];

p2 = new char[60];

p3 = new float[8];

delete [] p2;

p2 = new char[70];

//////// Problem 3.8 ////////////////

