ICS 332 Fall 2021

Homework Assignment #9 – Main Memory [40 pts]

You are expected to do your own work on all homework assignments. (See the statement of Academic Dishonesty on the Syllabus.)

Check the Syllabus for the late assignment policy for this course.

How to turn in?

Assignments need to be turned in via Laulima. Check the Syllabus for the late assignment policy for the course.

What to turn in?

You should turn in single **plain text** file named `README.txt` with your answers to the assignment's questions. Your file must be readable "as is" and points will be removed if the report is not readable.

Exercise #1 [40 pts]

In this exercise we "reverse engineer" code to try to determine how the heap in a C program is managed. **You do not need to compile and/or run this program**. Consider the following C program (compiled as an executable called `memory`):

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
   int chunk_sizes[4];


   if ((argc != 2) ||
       (sscanf(argv[1], "%d,%d,%d,%d", &chunk_sizes[0], &chunk_sizes[1], &chunk_sizes[2], 
       (chunk_sizes[0] < 0) || (chunk_sizes[1] < 0) || (chunk_sizes[2] < 0) || (chunk_size
      fprintf(stderr,"Usage: %s < a , b, c, d>\n", argv[0]);
      fprintf(stderr,"       where a, b, c, and d above must be >=0 integers (chunk sizes)
      exit(1);
   }

   char *ptrs[9];
   unsigned long sizes[9] = {1024, 64, 1024, 512, 1024, 128, 1024, 64, 1024};
```

ICS 332 Fall 2021

```
   }
   printf("\n--------------\n");

   for (int i=1; i < 9; i += 2) {
      free(ptrs[i]);
   }

   for (int i=0; i < 4; i++) {
          if (chunk_sizes[i] > 0) {
                char *chunk = (char *)calloc(chunk_sizes[i], sizeof(char));
                printf("The %d-byte chunk was allocated at address %ld\n", chunk_sizes[i],
          }
   }
   exit(0);
}
```

On some system, invoking the program as

```
./memory  0,0,0,0
```

produces this output:

```
0 | 1024 | 1088 | 2112 | 2624 | 3648 | 3776 | 4800 | 4864 |
--------------
```

Question #1 [4 pts]

If invoking the program as

```
./memory  480,0,0,0
```

produced this output:

```
0 | 1024 | 1088 | 2112 | 2624 | 3648 | 3776 | 4800 | 4864 |
--------------
The 480-byte chunk was allocated at address 2144
```

would you conclude that the heap allocator allocates new chunks of RAM at the beginning (i.e., lower addresses) or at the end (i.e., higher addresses) of holes in memory? Why? To explain why just use a contraposition argument (e.g., if the heap allocator did X then Y would have happened, but instead Z happened).

Whatever the heap allocator does, this is what we assume it does for all subsequent questions. **Also, in all subsequent questions, we assume that all considered algorithms**

ICS 332 Fall 2021

Question #2 [10 pts]

If invoking the program as

```
./memory  100,200,100,80
```

produces this output:

```
0 | 1024 | 1088 | 2112 | 2624 | 3648 | 3776 | 4800 | 4864 |
--------------
The 100-byte chunk was allocated at address 2524
The 200-byte chunk was allocated at address 2324
The 100-byte chunk was allocated at address 2224
The 80-byte chunk was allocated at address 3696
```

would you say that the heap allocator uses

- first fit
- best fit
- worst fit
- none of the above?

Question #3 [10 pts]

If invoking the program as

```
./memory  40, 400, 80, 60
```

produces this output:

```
0 | 1024 | 1088 | 2112 | 2624 | 3648 | 3776 | 4800 | 4864 |
--------------
The 40-byte chunk was allocated at address 3736
The 400-byte chunk was allocated at address 2224
The 80-byte chunk was allocated at address 2144
The 60-byte chunk was allocated at address 4804
```

would you say that the heap allocator uses

- first fit
- best fit
- worst fit
- none of the above?

Question #4 [16 pts]

ICS 332 Fall 2021

```
./memory  400, 100, 10, 10
```