Homework 1

MPCS 51100 Advanced Programming

Yan Cui

Compile

- All source files can be complied by 'make'

P1

- Bitwise shifting in C arithmetic shift

P2

- Let a pointer of integer point to the address of the input float, then use dereference operator and bitwise operators to get bit patterns.
- The bit pattern of the max float should be: 0 11111111 11111111111111111111111 that means: 1.11111111111111111111 << 128, equals to 3.40282e+38, the min value should be -3.40282e+38. The it pattern of the min absolute value should be: 0 00000001 00000000000000000000000, that is 1 >> 127, equals to 1.17549e-38.

P3

- Heapify referenced from: https://en.wikipedia.org/wiki/Binary_heap

P4

- Firstly allocate memory to every double**, then allocate memory to double*;

P5

- Allocate (sizeof(double*) m + sizeof(double) nm) bytes to the pointer, since we need sizeof(double*) * m to store double* and sizeof(double) * nm to store all the value, since we only use malloc once, all the memory are contiguous. We only need to free the memory once.

P6

- Similar to P5 while we have to allocate (sizeof(double**) * l sizeof(double*) * ml +

sizeof(double) * nml) to store all the pointers and values

P7

```
A Statically allocated 2D array
B Dynamically allocated 2D array, contiguous in memory
C Dynamically allocated 2D array, distributed through memory
```

- When n = m = 1000, A has the best performance and B,C proforms equally. When the 2D array is small they can be stored into cache, the differences between their profmorance may be reduced.
- When n = m = 10000, the runtime is A:0.457721 B:0.504609 C:0.526687. The statically allocated 2D array is still the most efficient way.

P8

- Use getline() to get names from stdin and quick sort (referenced from https://en.wikipedia.org/wiki/Quicksort) to alphabetizes lines.

P9

- Use getline() and file pointer to read a file and store the contents.

P10

- Use an array to record every words and their occurrences. When a new word with a repeated letter comes, check if it's already in the array. If it is, increase its occurrence by one, otherwise insert the word to the end of the list.