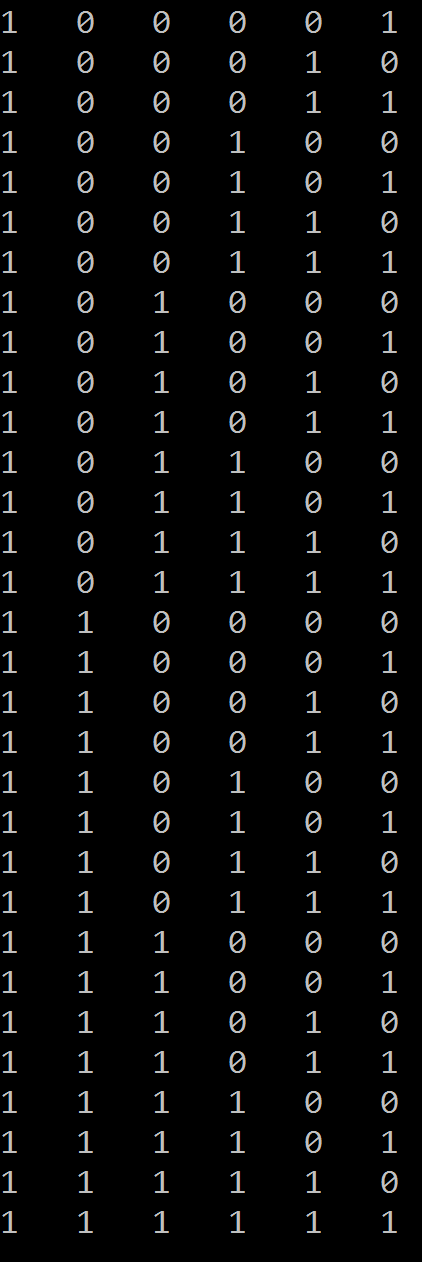
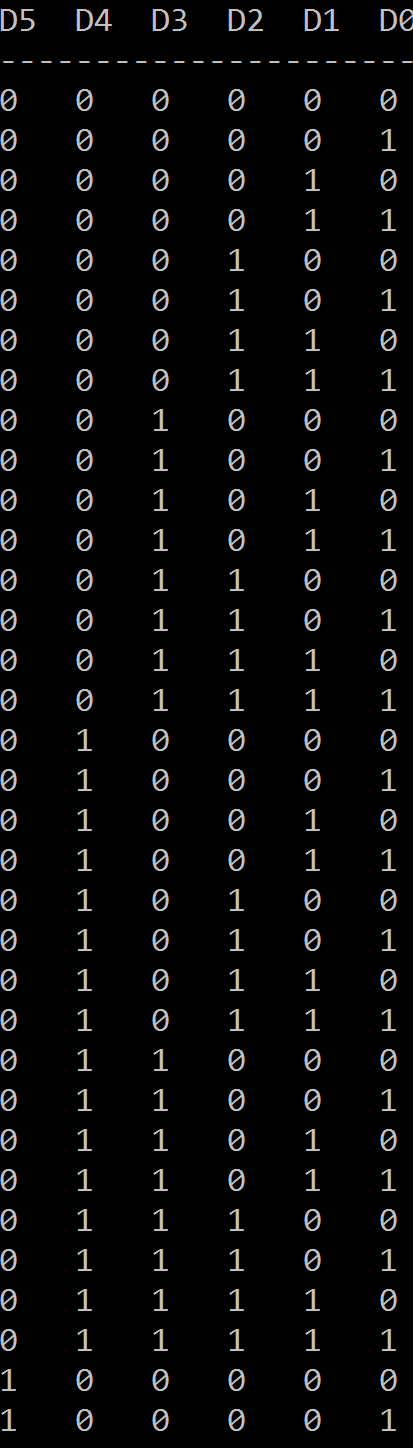
Melinda Ong

CPE 329-01

Assignment 0: C Programming Boot

1. 8-bit binary counter command prompt output  
     
   
2. 6-bit Counter Source Code

#include <stdio.h>

#include <string.h>

int main()

{

// Print header

printf("D5 D4 D3 D2 D1 D0\n");

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

int count;

// Print bit counter

for (count = 0; count <64; count++)

{

// For each bit place, prints appropriate '0' or '1' according for 0 to 63

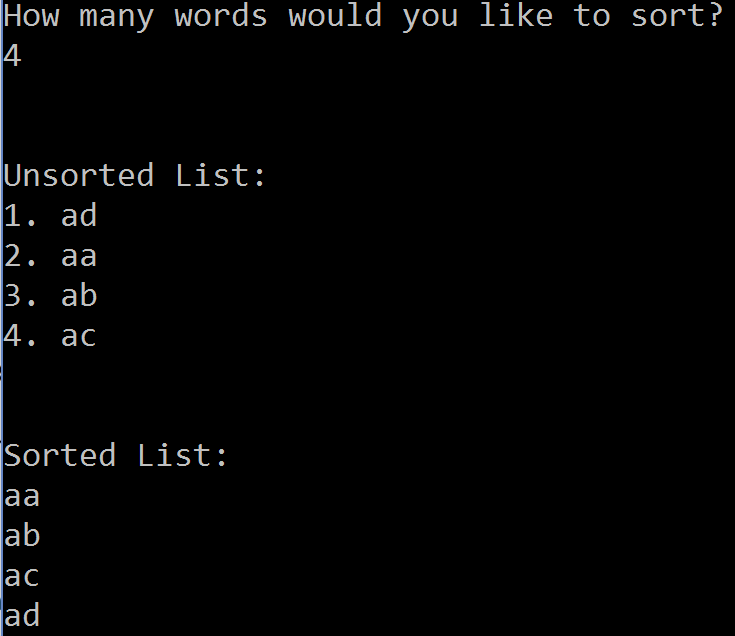
printf("%d %d %d %d %d %d\n", (count & 32) >> 5, (count & 16) >> 4,

(count & 8) >> 3, (count & 4) >> 2, (count & 2) >> 1, (count & 1));

}

return 0;

}

1. Word sort command prompt output  
   
2. Word sort source code

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

// Method for comparing two words

int wrdcmp(const void\* word1, const void\* word2)

{

return strcmp(\*((char\*\*)word1), \*((char\*\*)word2));

}

int main()

{

int count;

char word[256], temp[256];

// Prompts user for number of words to sort

printf("How many words would you like to sort?\n");

scanf("%s", temp);

// Converts temp char to int

count = atoi(temp);

// Allocates memory in string array

char\*\* words = malloc(sizeof(char\*)\*count);

// Prompts user for unsorted words

printf("\n\nUnsorted List:\n");

for (int i = 0; i < count; i++)

{

printf("%d. ", i+1);

scanf("%s", word);

int len = strlen(word);

words[i] = malloc(sizeof(char\*)\*(len+1));

strcpy(words[i], word);

}

// Sorts unsorted words

qsort(words, count, sizeof(char\*), wrdcmp);

// Prints sorted list of words

printf("\n\nSorted List:");

for (int k = 0; k < count; k++)

{

printf("\n%s", words[k]);

}

}