**Problem 1. (40 points)**

Using signed representation and size of 6 bits, show the addition in binary of 16 + 24, then convert the result from binary to decimal and explain why you obtained that result.

010000 16

+ 011000 +24

**101000 -24**

The result is -24 because in a 6-bit size, the highest binary representable number in the range is 31. However, if we were to add 16 + 24 normally, the decimal result is 40, which falls outside of that range. Therefore, this leads to an overflow and causes the result to wrap around. Since the result is in signed representation, the leftmost bit also indicates the sign of the number (twos-complement). In this case, it is a 1, so the result is interpreted as negative. Finding the two’s complement of the binary value yields a result of -24.

**Problem 2. (60 points)**

Start with the code (shown below) from Example 1 in file Examples-in-C.pdf. The original code prints the contents of a text file line-by-line but backwards so that the last line is printed first. The filename is specified as a command-line argument.

Write an improved version of the code to do the following:

• Change the linked list to be doubly-linked: each node X has a pointer to the node after node X and to the node before node X. A pointer root points to the head of the queue and a pointer end points to the tail of the queue.

• Every new line read from the file is stored in a new node which is inserted at the head of the linked list (as in the original code). Pointer root points to the node with the last line read so far.

• After the whole file is read, print only the lines that contain the word “test” (without the quotes) in the order that the lines appear in the file. The output should be the same as the output of grep test filename.

• If you use functions from the C library, make sure you include a short description of what each one of these functions does. Otherwise, you should include the code for any additional function you chose to call (if they are not part of the C library).

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node {

char info[256];

struct node \* prev; // Doubly linked list has a reference to the previous node

struct node \* next;

};

int main (int argc, char \* argv[]) {

FILE \* fp;

char line[256];

struct node \* root;

struct node \* end; // Declare pointer to the end of the queue

struct node \* new\_node;

struct node \* p, \*q;

if (argc == 1) {

printf ("Error, no file provided.\n") ;

return(0);

}

fp = fopen (argv[1], "r") ;

if (fp == NULL) {

printf ("Error, file not found.\n") ;

return(0);

}

root = NULL;

end = NULL; // Initialize the end pointer to be null

while (fgets (line, 256, fp) != NULL) {

new\_node = malloc (sizeof (struct node) ) ;

strcpy (new\_node -> info, line);

new\_node -> next = root;

// Nothing to point to previously since there are no nodes before the head

new\_node -> prev = NULL;

if (root) { // Update previous pointer if the list is not empty

root->prev = new\_node;

}

root = new\_node;

if (! end) { // First node is also the end of the queue (initializes at the beginning)

end = new\_node;

}

}

for (p = root; p != NULL; p = p -> next) {

// We need to only print if the line contains the word test

if (strstr(p->info, "test")) {

printf ("%s", p->info) ;

}

}

for (p = root; p != NULL; p = q) {

q = p -> next;

free(p);

}

fclose(fp); // Close file when done

return(0);

}