CS 100 Final Exam – Fall 2017 – Coding

Instructions

- Create a directory called **final**
- Complete the three programs shown below.
- When you are finished, compress your **final** directory into a single (compressed) file.
- Submit your file, named **final.zip**, to Blackboard.
- You cannot use the Internet when coding these three problems.
- You can log into the **cs-intro.ua.edu** server to test your programs.

---

1. Write a program named **one.c** that takes a single command-line argument, the name of a file. Your program should read all the strings (tokens) from this file and write all the strings that are potentially legal integers (contains only digits and a possible leading negative sign) to the file **integers**. Your program should ignore everything else (do not write those strings anywhere)

   As an example, running **./a.out dataFile** would result in the generation of the file **integers** shown at the right below.

dataFile	integers
2 4 6.8 cat -1831 -1-2-3 #RollTide 100 -101 12-15-2017	2 4 -1831 100 -101

   You should print an appropriate error message if the input file specified does not exist. You can assume that none of the strings (tokens) in the input file are longer than 100 characters.

---

2. Complete a program named **two.c** that manipulates linked lists. A template for the program is shown on the next page and can be downloaded from Blackboard. The template adds items to the list (adding new items at the front). After creating the list, the main routine then prompts the user for a name and prints all the names that occur in the list **after the specified name**. For example, if the linked list contains the names

   Betty Barney Wilma Fred

   then when the user specifies "**Barney**" the program should print the names that occur after **Barney** (**Wilma** and **Fred**).

   If the name does not exist in the list, print nothing.

---

3. Write a program named **three.c** that checks whether or not a given row of a two-dimensional matrix has any duplicate values. A template for the program is shown on the next page and can be downloaded from Blackboard. The program allocates space for a square two-dimensional matrix and then reads the matrix from the specified file.

   After initializing the matrix, the program prompts the user for a row number and determines whether or not the row in question has any duplicates.

   For example, given that the file **data** contains the data shown at the right, then the program executes as follows:

1	2	3	4	1
6	7	8	9	10
11	12	13	12	11
16	17	18	19	20
21	21	21	21	21

```
./a.out data 5
Enter the row: 0
Row 0 has duplicates

./a.out data 5
Enter the row: 3
No duplicates in row 3
```

   You do not have to check for illegal input, all command-line arguments and input values specified for the row will be legal values.

Code for **two.c** (can be downloaded from Blackboard)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct node {
    char *name;
    struct node *next;
} Node;

void printAfter(Node *, char *);

Node *add(Node *ptr, char *name) {
    Node *newOne = malloc( sizeof(Node) );
    newOne->name = malloc( strlen(name) + 1 );
    strcpy(newOne->name, name);
    newOne->next = ptr;
    return newOne;
}

int main(void) {
    char str[100];
    Node *myList = NULL;
    printf("Enter a name to add : ");
    scanf("%s", str);
    while (strcmp(str, "quit") != 0) {
        myList = add(myList, str);
        printf("Enter a name or \"quit\" : ");
        scanf("%s", str);
    }
    printf("\n\n\nEnter a name : ");
    scanf("%s", str);
    printf("The list after %s\n", str);
    printAfter(myList, str);
    return 0;
}
```

Code for **three.c** (can be downloaded from Blackboard)

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

int hasDups(int **, int, int);

int main(int argc, char *argv[]) {
    FILE *fp = fopen(argv[1], "r");
    int size = atoi(argv[2]);

    // allocate the matrix
    int **matrix;
    matrix = malloc( sizeof(int *) * size );
    for (int a=0; a<size; a++)
        matrix[a] = malloc(sizeof(int) * size);

    // read the matrix
    for (int a=0; a<size; a++)
        for (int b=0; b<size; b++)
            fscanf(fp, "%d", &matrix[a][b]);

    int row;
    printf("Enter the row : ");
    scanf("%d", &row);

    // compute the answer
    int ans = hasDups(matrix, size, row);
    if (ans)
       printf("Row %d has duplicates\n", row);
    else
       printf("No dupicates in row %d\n", row);
    return 0;
}
```