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 words (the string contains only upper-case and lower-case characters in any combination) to the file **words**. 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 **words** shown at the right below.

dataFile	words
Crimson Tide, the U-of-A, #1 since 1831 Also, CS is great!	Crimson the since CS is

   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 program 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 **before the specified name**. For example, if the linked list contains the names

   Betty->Barney->Wilma->Fred

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

   If the name does not exist in the list, print the entire list.

---

3. Write a program named **three.c** that computes the average for a subset of a two-dimensional matrix. 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.

   Your program then computes the average value for the subset of the matrix specified by the two coordinates.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	10

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

```
./a.out data 5
Enter coordinates: 1 1 2 3
Average: 10.500000
```

   In this example, you are computing the average of the elements shaded in yellow, which is **7 + 8 + 9 + 12 + 13 + 14 is 63 / 6 = 10.5**

   The coordinates are entered in the format of **r1 c1** then **r2 c2**

   You can assume that all command-line arguments are legal and all coordinates represent legal positions in the matrix. You do not have to check for illegal input.

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 printBefore(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 before %s\n", str);
    printBefore(myList, str);
    return 0;
}
```

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

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

double blockAvg(int **, int, int,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]);

    // compute the answer
    printf("Enter coordinates : ");
    int x1, y1, x2, y2;
    scanf("%d %d %d%d", &x1, &y1, &x2, &y2);
    double answer =
        blockAvg(matrix, size, x1, y1, x2, y2);

    printf("Average: %lf\n", answer);
    return 0;
}
```