BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)

Second Semester 2017-18 CS F111 Computer Programming

LABORATORY SESSION #10 SOLUTIONS

(Dynamic Memory Allocation; Array of Pointers)

1. Declare an array of **m** pointers, each pointing to a dynamically allocated array of **n** integers (m and n are both user inputs). Then, take m x n inputs for the matrix elements. Next, display the entire matrix on screen in a neatly formatted manner. If you wish, you can use the data file `/home/share/10.1.txt` for taking inputs (first line contains m and n, and the remaining lines have the values for the m x n matrix).

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

int main()
{
  int m, n, i, j;
  scanf("%d", &m);
  int *mat[m];        // this is a VLA, allocated in stack memory
   scanf("%d", &n);
  for (i = 0; i < m; ++i)
    mat[i] = (int *) calloc(n,sizeof(int)); // heap memory

   // Henceforward, mat can be used exactly like a 2D array
  for (i = 0; i < m; ++i)
    for (j = 0; j < n; ++j)
          scanf("%d", &mat[i][j]);

  for (i = 0; i < m; ++i)
  {
    for (j = 0; j < n; ++j)
        printf("%7d", mat[i][j]);
    free(mat[i]);// good practice to free heap space
    putchar('\n');
  }
  return 0;
}
```

2. This program involves taking input for m rows of a matrix, but each row having a varying number of integer elements. Take a look at `/home/share/10.2.txt` (and copy to your current directory) to get an idea of this matrix with varying number of columns for each row. The first row contains the number of rows. The first element of each row contains the number of columns of that particular row, followed by the actual elements themselves.

a. Now declare a suitable array of pointers, dynamically allocate memory for each row, read the elements (including the number of elements as the first entry of each row), and then display the entire matrix on the screen. Verify to make sure

that your output is the same as the matrix entries given in the data file you copied (`/home/share/10.2.txt`).

b. Take a look at the ATM branches problem you worked on during last week's lab (question #3 of Lab Sheet #9). Copy into your current working directory the C program you wrote last week for this question and rename the file as `10.2.c`. Now consider the following modification of the problem statement:

> *It was observed that there were few ATM branches that had very few transactions, but others that had several. The file available at `/home/share/10.2.txt` has the data – the first row has the number of branches, while the first number in each of the other rows (representing a branch) indicate the number of transactions recorded for that branch, followed by the actual transaction values.*

Now modify the code in `10.2.c` in a way the new data can be captured – using a dynamically allocated array rather than an m x n matrix allocation you had done earlier.

```
#include <stdio.h>
#include <stdlib.h>   /* for using malloc(), calloc() and free() */

int main()
{
  int no_branches, no_transactions, i, j;
  scanf("%d", &no_branches);
  int *mat[no_branches];
  for (i = 0; i < no_branches; ++i)
  {
        scanf("%d", &no_transactions);
        mat[i] = (int *) calloc(no_transactions+1,sizeof(int));
        mat[i][0] = no_transactions;
        for (j = 1; j < no_transactions+1; ++j)
          scanf("%d", &mat[i][j]);
  }
  for (i = 0; i < no_branches; ++i)
  {
    no_transactions = mat[i][0];
    printf("%7d", mat[i][0]);
    for (j = 1; j < no_transactions+1; ++j)
        printf("%7d", mat[i][j]);
    free(mat[i]);
    putchar('\n');
  }
  return 0;
} /* end of main() */
```

3. Two words/phrases are said to be anagrams if one of them can be rearranged to give the other. Write a function **`isAnagram()`** that takes two strings as arguments and tells

whether the second string is an anagram of the second or not. For example, the function should return a value of 1 for pair of string inputs such as “great” and “grate”; “Listen” and “siLent”; “rail safety” and “fairy tales”. It returns a 0 for string inputs such as: “night” and “knight”; “great” and “Grate”; “Madam Curie” and “Radium came”. [Hint: One of the many strategies that can be used to solve this problem involves sorting the individual characters of the string! If you need to use a sorting function, you may use the code for selection sort available at `/home/share/10.3.c` in your program.]

```
int isAnagram(char *str1, char *str2)
{
char w1[50], w2[50];

if (strlen(str1) != strlen(str2))
return 0;    /* strings of unequal length can't be anagrams */
if (strcmp(str1,str2) == 0)
return 1;
strcpy(w1,str1);
strcpy(w2,str2);
selectionSort(w1,strlen(w1)); /* sorting the letters of the first
                                 string */
selectionSort(w2,strlen(w2)); /* sorting the letters of the second
                                 string */
if (strcmp(w1,w2) == 0)    /* sorted strings are the same */
   return 1;
else
   return 0;
}

void selectionSort(char s[], int len)
{
int i, j;
char ch;
for (i = 0; i < len-1; ++i)
   for (j = i+1; j < len; ++j)
        if (s[j] < s[i])
{ ch = s[j]; s[j] = s[i]; s[i] = ch; }    // swapping
return;
}
```

4. In the class, we discussed selection sort technique that sorts an array of elements. Insertion sort is another technique for sorting whose algorithm is given below:

```
1. Read N, Arr
2. FOR i = 1 to N-1
        Key = Arr[i]
        j = i - 1
        WHILE ( j >= 0 and Arr[j] > Key)
              Arr[j+1] = Arr[j]
              j = j - 1
```

```
            Arr[j+1] = Key
        3. Output Arr
        4. Stop
```

a. Understand the algorithm by taking a sample array of elements {8,5,3,9,4}. Write down in your lab notebook the stepwise results of applying the above algorithm on this input.

b. Translate this algorithm into a C function which has the following prototype:

```
void insertionSort(int arr[], int num);
```

Test the function by calling it in `main()` with sample data.

```
void insertionSort(int arr[], int num)
{
  int i, key, j;
  for (i = 1; i <= num - 1; ++i)
  {
    key = arr[i];
    j = i - 1;
    while (j >= 0 && arr[j] > key)
    {
        arr[j+1] = arr[j];
        j = j - 1;
    }
    arr[j+1] = key;
  }
}
```