Map A

Note: try not to use the internet and instead use only the man pages. If you really must use the internet, don’t look for code samples that implement your task (or a similar one) but rather look for code samples that implement a specific “small” task (i.e. how to print a string to the console etc.).

Arrays:

Here is a function to reverse the values in the array:

```
#include < stdio.h >
#define ARRAY_SIZE 10

void reverse(int Array[ARRAY_SIZE]);

int main()
{
  int i, Array[ARRAY_SIZE];
  printf(" Enter elements into the Array :n ");
  for (i=0; i< ARRAY_SIZE ; i++)
  {
    printf("Enter %2d elements : ",i);
    scanf("%d",&Array[i]);
  }
  printf("n***********************nn");
  reverse(Array);
  printf("***********************nn");
  printf(" here is the new Array : n ");
  for (i=0; i< ARRAY_SIZE ; i++)
  {
    printf("%4d",Array[i]);
  }
  return 0;
}

void reverse(int Array[ARRAY_SIZE])
{
  int temp,i;
  for ( i=0; i < (ARRAY_SIZE/2) ; i++)
  {
    temp = Array[ARRAY_SIZE-1-i];
    Array[ARRAY_SIZE-1-i] = Array[i];
    Array[i] =temp;
  }
}
```

1. Write a function that accepts an array and sorts it (by insertion sort, or by any sorting algorithm you see fit) in descending order.
   For example I received an array of:
   [1,6,3,9,4] => [9,6,4,3,1]

Strings:
Here is a program that accepts a string (an array of characters) and returns the reverse string:

```
#include <stdio.h>
void reverse (char s[ ] )

/* reverse character string */
{
  int string_length;
  int i;
  char tmp;

  /* step 1: calculate length of s */
  string_length = 0;
  while ( s[string_length] != '\0')
    string_length = string_length + 1;

  /* step 2: reverse string */
  for (i = 0; i < string_length/2; i++)
  {
    tmp = s[i];
    s[i] = s[string_length - i - 1];
    s[string_length  i  1] = tmp;
  }
}
```

2.
   a. Write a program that checks if a given password (array of chars) is "strong".
      A strong password is a password:
      a) whose length is at least 5 characters.
      b) which contains at least one number.
      c) that contains at least one capital letter.
      d) which contains at least one of the following symbols %!#@
      Print whether the password is strong or not.

   b. Write a function called expand(s1) which expands
      Abbreviations such as a-z in string s1 and prints the full list equivalent to: abc…xyz.
      Upper and lower case letters, as well as numbers, should be allowed.
      Let's assume that the string has a valid syntax, and only expressions of the form a-z 0-9 appear in it

Recursion:
A string of characters is called a **“palindrome”** if we get the same result when reading from right to left and from left to right.
Write code that checks if the string is a palindrome. For example, the following string is a palindrome: “abba”

```
int isPalindrome(char str[], int n, int startFrom)
{
  int checkIfPalindrome = 1;
  if ((checkIfPalindrome) && (n != startFrom))
  {
    if (str[startFrom] == str[n])
    {
      checkIfPalindrome = isPalindrome(str, n - 1, startFrom + 1);
    }
    else {
      checkIfPalindrome = 0;
    }
  }
  return (checkIfPalindrome);
}
```

3.
    a. Advanced definition of palindrome: all characters which are not letters should be skipped, and the check is not case sensitive, i.e., upper case letters are assumed equal to lowercase ones.
    For example, the following string is a palindrome in the advanced definition: “Madam! I’m Adam”
    Write code that checks if the string is a palindrome according to the advanced definition.

    b. A quick sort algorithm is a recursive algorithm that works with the divide-and-conquer method. His steps are as follows:

        A. Given a series of members, choose a member from the series at random (called: pivot, or "axis member").
        B. Arrange all the members so that the members larger than the pivot member appear after the members smaller than the pivot member.
        C. Recursively, run the algorithm on the set of larger terms and the set of smaller terms.
        D. The stopping condition of the algorithm is when there is one member, then the algorithm announces that the series is sorted.

    Write a program to implement quick sort:
    Here is a partial implementation containing the functions: init (to create and fill the array) and print (to print an array).
    Please fill in the missing functions to complete the algorithm.

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

int SIZE = 30;
int MAX = 1000000;
int array[SIZE];

void init() {
  int i;
  for(i=0; i<SIZE; ++i)
    array[i] = random()%SIZE+1;
}

void printArray() {
  int i;
  for(i=0; i<SIZE; ++i)
    printf("%d, ",array[i]);
  printf("\n");
}
```

pointers:

Here is a function that swaps the values of two variables:

```
void swap(int *a, int *b) {
    int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
}
int main(){
    int a = 3;
    int b = 7;
    swap(&a, &b);
}
```

Here is a function that accepts an array of n pointers to array and return an array with sum of all elements in each array

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

int* sum_of_arrays(int** a, int n)
{
    int* result = malloc(4*sizeof(int));
    for (int i = 0; i < n; ++i){
        int sum = 0;
        for (int j = 0; j < n; ++j){
            sum += *(*(a + i) + j);
        }
        result[i] = sum;
        sum = 0;
    }
    return result;
}
```

```
int main()
{
    #define SIZE 4
    // creating matrix of 4x4
    int* a[SIZE];
    int b[SIZE] = {1,2,3,4};
    int c[SIZE] = {6,4,8,3};
    int d[SIZE] = {5,9,22,5};
    int e[SIZE] = {10,5,6,23};

    *a = b;
    *(a+1) = c;
    *(a+2) = d;
    *(a+3) = e;

    int* r = sum_of_arrays(a, SIZE);

    for (int i = 0; i < SIZE; ++i){
        printf("%d \n", *(r + i));
    }

    free(r);
}
```

4.
    a. Given two pointers of arrays a and b:
       int arr_a[ ] = {5,3,2,6};
       int arr_b[ ] = {6,9,2,5,7,3};
       int* a = arr_a;
       int* b = arr_b;
       Write a function that swap the arrays of the two pointers.

    b. Write a function that accepts two variables, and after calling, the first variable contains the sum of two variables, and the second variable contains the difference between them.

5. Given a pointer to the head node of a linked list, the task is to reverse the linked list. We need to reverse the list by changing the links between nodes.

   For example:
   Input: Head of following linked list
   1->2->3->4->NULL
   Output: Linked list should be changed to,
   4->3->2->1->NULL

   You have linked list, add function - static void reverse(struct Node** head_ref)

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

// Link list node
struct Node {
	int data;
	struct Node* next;
};

// Function to add a new node
void insert(struct Node** head_ref, int new_data)
{
	struct Node* new_node
		= (struct Node*)malloc(sizeof(struct Node));
	new_node->data = new_data;
	new_node->next = (*head_ref);
	(*head_ref) = new_node;
}

// Function to print linked list
void printList(struct Node* head)
{
	struct Node* temp = head;
	while (temp != NULL) {
		printf("%d ", temp->data);
		temp = temp->next;
	}
}


int main()
{
	// Start with the empty list
	struct Node* head = NULL;

	insert(&head, 20);
	insert(&head, 4);
	insert(&head, 15);
	insert(&head, 85);

	printf("The linked list is:\n");
	printList(head);
}
```