Topic: Pointers

Practice Problems: Set 1

1. Implement bubble sort algorithm. Write a function *bubbleSort(int * items, int n, int * comp)* that will receive an integer array (*items*), the size of the array (*n*), and another integer pointer *comp* as parameters. The function will sort the integers in ascending order. The function will also count the number of comparisons required to sort the array, and write the value in the memory address of *comp* pointer. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

2. Implement merge algorithm to merge to sorted arrays into a single sorted array. Write a function *merge(int * arr1, int n1, int *arr2, int n2, int *out)* that takes two sorted arrays as input and another pointer to write the output. The function will generate a merged sorted array and write it in the memory pointed by *out*. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

3. Function *strncpy(char * dest, char * src, int n)* is equivalent to strcpy, except that a call to strncpy specifies the number of characters to copy from the source memory to the destination memory. Implement the function. Use pointer notation and arithmetic. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

4. Function *strcspn(char * src, char * search)* determines the length of the initial part of the string in its first argument *src* that does not contain any characters from the string in its second argument (*search*). The function returns the length of the segment. Implement the function. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

5. Function *strrchr(char * str, char ch)* searches for the last occurrence of a character (*ch*) in a string (*str*). If the character is found, *strrchr* returns a pointer to the character in *str*; otherwise, *strrchr* returns NULL. Implement the function. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

6. Write a function atoi*(char * str, int * num)* that converts the integer value stored as string (passed in parameter *str*) to a integer representation and write the output in the memory passed as *num.* Implement the function. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

7. Write a function *itoa(char * str, int * num)* that converts the integr value (passed in parameter *num*) to a string representation and write the output in the memory passed as *str.* Implement the

function. Use pointer notation and arithmetic. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

8. Write a function atof*(char * str, float * num)* that converts the floating point value stored as string (passed in parameter *str*) to a floating point representation and write the output in the memory passed as *num.* Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

9. Write a function *reverse(int * arr, int n)* that reverses the integer array passed as input. Implement the function. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

10. Write the function *strend(char * s,char * t)*, which returns 1 if the string *t* occurs at the end of the string *s*, and zero otherwise. Implement the function. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

11. Write a function maxCommSubArray(int *arr1, int n1, int * arr2, int n2, int * out, int * n3). The function computes the maximum length subarray common in both input arrays *arr1* and *arr2*. The function writes the result subarray in the memory pointed by *out* and the length of the subarray in the memory pointed by *n3*. Implement the function. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.

12. Write a function *sortNames(char * names[], int n)* to sort the names of students in alphabetical order. Use an array of pointers to store the names. Create memory location for the array of pointers using DMA in the main function, take inputs from user, and then call the *sortNames* function to sort the names in alphabetical order. Implement the function. Necessary memory allocations should be done from the main function for all arrays. Write a main function to demonstrate that your program generates output correctly.