Department of Computer Science and Engineering

Assignment 8

Subject : Programming and Data Structure (CS19003)

Instructions:

- Give the name of the programs files as <Your roll>_<assignment number>_<question number>.c. For example, 21XX12345_A1_Q1.c for question 1 of assignment 1 of a student with roll 21XX12345. **Follow the file naming convention strictly**.
- Apart from the main .c file for each program, you should also upload one additional temporary .c file for each program (such as when you have finished half of the code). The naming for the temporary file should be in the format <Your roll>_<assignment number>_<question number>_temp.c. For example, 21XX12345_A1_Q1_temp.c **Make sure that your main code do not deviate much from its temporary code for each program**.
- You should upload the main .c file and the temporary .c file individually to the Moodle course web page once you finish answering each question. No need to zip the files.
- The **deadline** to upload the programs is 12:00PM. Beyond that, submission will be closed (No extensions will be granted).
- If you do not follow the instructions, your marks may be deducted.

[Total Marks = 100 (30 + 30 + 40)]

Answer all questions

1. Take input a long sentence that contains an arbitrary number of words. Two words are supposed to be separated by only one blank space character. Also a valid word consists of only the ASCII characters 'A-Z' or 'a-z' and nothing else. However the sentence that you will be taking as input may not follow this rule, i.e there may be more than one blank spaces between two words as well the words themselves may contain invalid characters. Your task is to refine this invalid input to get the correct valid string **in place**, i.e that you CANNOT use any additional array to perform this operation, you have to process within the same input array with perhaps one or two additional character variables. Print the final valid string as well. You can assume that any sequence of characters 'A-Z' or 'a-z' forms a valid word although it may not exist in the dictionary [No need to consider full stop also at the end]. You are **NOT** allowed to use any in-built string functions.

   Example:

   Input:

   Acceler56ation is def9ine%d as r34@at56e 78of chang4e of ve2@locity

   Output:

   Acceleration is defined as rate of change of velocity **[30]**

2. Camel case notation is a special type of typography where multiple different words are combined together as one word without any space in between AND the first letter of each different word is uppercase while the remaining letters are lowercase. For example 'MySchool' is a camel case notation consisting of two separate words 'my' and 'school', while 'IAmHappy' is a camel case notation containing three words, 'I', 'am' and 'happy'. Now consider the following two functions to perform the tasks as mentioned.

   - **int numberOfWords(char *camel)** - takes input a camel case string char *camel as argument and returns the number of constituent words.
   - **void camelToReverse(char *camel, char *reverse)** - Takes input two arguments, one camel case string (char *camel) and another empty string (char *reverse). Hence it fills up the empty string with the constituent words of the camel case string **in the reverse order** (as it appears in the camel case string) with a space in between the constituent words and terminated with null. For example if the camel case string is 'IAmHappy', then reverse[ ] will contain 'Happy Am I', which is also a null terminated string.

   Write a C program to input a camel case string containing any number of constituent words. First print the number of constituent words of the input string by calling the appropriate function within main(). Then call the function camelToReverse() to generate the reverse string of the constituent words and print them within that function only. You can assume that the input is always a valid camel case string. Note that you can declare character arrays inside main() for taking the string input and initializing the empty string, BUT you must work with pointers to string within the two functions, i.e while calling the functions you must pass the base address of the character arrays and work with pointers inside the respective functions. You are **NOT** allowed to use any in-built string functions except probably strlen() **ONLY**.

   Example:

   Input:

   'HowDoYouDo'

   Output:

   Number of words: 4

   Reverse string: 'Do You Do How' [using pointers]

   [NOTE the space in between the constituent words] **[30]**

3. It is unsafe to send data directly from one place to another. Therefore, we use encryption to convert the files/strings into an unreadable format which needs some key/ method to decrypt to get the actual file/string. Write a C program to encrypt a text string following the given steps:

   1. Take a string as input (the string can contain alphabets and numbers)

   2. Take the 8-bit binary representation of each character in the input string (you can use int $i = (int)c$; if c is a char) and concatenate all these bits to get an intermediate representation of the whole input string.

   3. Then take each groups of 6-bits in the intermediate representation (starting from left-end towards the right) and map it using the following rules:

   $0 => A, 1 => B, ..., 25 => Z$ (Capital Letters)

   $26 => a, 27 => b, ..., 51 => z$ (Small Letters)

$52 => 0, 53 => 1, ..., 61 => 9$ (Numericals)

Example:

Input:

IIT Kharagpur

[Intermediate steps:

1. (int) 'I' is 73
2. 8-bit binary representation of 73 is 01001001
3. Hence, intermediate representation is 0100100101001001...
4. After converting first 6-bits from the intermediate representation, we get $(010010)2 => (18)10 => S$]

Output:

SUlUIEtoYXJhZ3B1cC

Input:
C Programming

Output:

QyBQcm9ncmFtbWluZC

[Hint: You can consider a temporary integer array of size (8*number of characters of string) where you can store the intermediate bit representation, each bit as an integer 1 or integer 0 stored in each cell of the array] **[40]**
