Programming in C: Prerequisite to Data Structures
Additional Assignment - I

1. Write the following programs without using the array indexing operator (`[...]`). If you have difficulty, first write programs using the array indexing operator (`[...]`), and then, attempt to rewrite them without using the operator. Do not use the library functions in `string.h`.
   - ✓ (a) A function to find the length of a string.
   - ✓ (b) A function to compare two strings. If the contents of the two strings are the same, it should return 1; otherwise, it should return 0.
   - ✓ (c) A function to check if two strings are the same. The function should return 1 if they are the same; otherwise, it should return 0.
   - ✓ (d) A function to search for the first occurrence of a given character (an unsigned char) in the first n bytes of a given string.
   - ✓ (e) A function to append a given string to the end of another given string.
   - (f) A function to calculates the length of the initial segment of a given string which consists entirely of characters, not in another given string.
   - (g) A function to finds the first character in a given string that matches any character specified in another given string.
   - (h) A function to search for the last occurrence of a given character (an unsigned char) in a given string.
   - (i) A function to reverse a string without using another string.
2. Create an array of integers with dynamic memory allocation. The size of the array is user input. Read the elements of the array as user inputs.
   - (a) Write a function to perform insertion sort on that array.
   - (b) Write a function to perform bubble sort on that array.
   - (c) Write a function to perform selection sort on that array.

   Next, rewrite these programs without using the array indexing operator (`[...]`).
3. Write a function to allocate a three dimensional matrix, such that, its ith element in the first dimension is a two-dimensional matrix with $g(i)$ number of rows and $h(i)$ number of columns, where $g(i) = i + 1$ and $h(i) = (i + 2) \times g(i)$ (i begins with 0). Use this function to create a three dimensional matrix with 3 elements in the first dimension. Write another function to make this allocated memory free. Next, rewrite these programs without using the array indexing operator (`[...]`).
4. Write a program to convert a real value (double) in $[-999999999, 999999999]$ to words. For instance, -50187035.57069 should be converted to the string "minus fifty crore eighteen lakh seven thousand thirty five point five seven zero six nine". Now, rewrite the program without using the array indexing operator (`[...]`).

5. Consider the following structures:

```
struct student {
    char *p_name;                // The name of the student
    int roll;                    // Roll no
    struct stream *p_stream;     // The associated stream
    char dummy;                  // To increase difficulty
};

struct stream {
    char *p_name;                // The name of the stream
    int code;                    // The code associated with the stream
    int num_subjects;            // The number of associated subjects
    struct subject *p_subjects;  // The set of associated subjects
    int num_students;            // The number of enrolled students
    struct student *p_students;  // The set of enrolled students
};

struct subject {
    char *p_name;                // The name of the subject
    int code;                    // The code associated with the subject
    int num_streams;             // The number of associated streams
    struct stream *p_streams;    // The set of associated streams
    int num_students;            // The number of enrolled students
    struct student *p_students;  // The set of enrolled students
};
```

Now, consider the following scenario. There are 5 students. The name and roll number of the ith student is "Name i" and $100 + i$, respectively (i begins with 0). The first three students belong to stream "CSE" (stream code 2) and the other two students belong to stream "ECE" (stream code 3). The stream "CSE" is associated with three subjects "Programming in C", "Data Structures", and "Operating Systems" with subject codes 1001, 1002, and 2001, respectively. Moreover, the stream "ECE" corresponds to three subjects "Programming in C", "Data Structures", and "Signal Processing" with subject codes 1001, 1002, and 3001, respectively.

Store this information in the memory such that, exactly five, two, and four instances of `struct student`, `struct stream`, and `struct subject` are created, respectively. You need to store the information in dynamically allocated memory such that the memory allocated for the student objects are contiguous. In the same fashion, the memory allocated for the stream objects and subject objects need to be contiguous. You are allowed to store only a pointer to the third student, i.e., the student with $i = 2$. You can store the literals in the data segment. Next, perform the following tasks:

(a) Print the subject codes of the subjects that are associated with the student 0.

(b) Print the name of the students who are associated with the first subject corresponding to the stream of the student 4.

(c) Say, the data structures used by you require the following amount of memory

$$\begin{aligned}
\mathcal{M}_{\texttt{total}} &= a_T \times \texttt{sizeof(int)} + b_T \times \texttt{sizeof(char)} + c_T \times \texttt{sizeof(void *)},\\
\mathcal{M}_{\texttt{heap}} &= a_H \times \texttt{sizeof(int)} + b_H \times \texttt{sizeof(char)} + c_H \times \texttt{sizeof(void *)},\\
\mathcal{M}_{\texttt{data}} &= a_D \times \texttt{sizeof(int)} + b_D \times \texttt{sizeof(char)} + c_D \times \texttt{sizeof(void *)},\\
\mathcal{M}_{\texttt{stack}} &= a_S \times \texttt{sizeof(int)} + b_S \times \texttt{sizeof(char)} + c_S \times \texttt{sizeof(void *)}.
\end{aligned}$$

Here, $\mathcal{M}_{\texttt{total}}$, $\mathcal{M}_{\texttt{heap}}$, $\mathcal{M}_{\texttt{data}}$, and $\mathcal{M}_{\texttt{stack}}$, respectively, denote the total memory allocated, the amount of the memory allocated in the heap, the amount of the memory allocated in the data segment, the amount of the memory allocated in the stack segment, respectively. What are the values of a_T, b_T, c_T, a_H, b_H, c_H, a_D, b_D, c_D, a_S, b_S, and c_S?

Now, rewrite the program twice. In the first case, you are not allowed to use the dot operator (`.`). In the second case, you are not allowed to use the arrow operator (`->`). Next, rewrite both of these programs without using the array indexing operator (`[...]`).

6. Is the following code a valid code? If it is a valid code, what is the output of the code assuming `&a = 0060FEFC`?

```
# include <stdio.h>

struct A {
    int data;
    struct B {
        int data;
        struct C {
            int data;
            struct D {
                int data;
            } d;
        } c;
    } b;
};

int main(int argc, char **argv) {
    struct A a = {'a', {'b', {'c', {'d'}}}};
    printf("%lu %c %c %c %c\n", sizeof(a), a.data, a.b.data, a.b.c.data, a.b.c.
        d.data);
    printf("%p %p %p %p %p\n", &a, &(a.data), &(a.b.data), &(a.b.c.data), &(a.b
        .c.d.data));
    return 0;
}
```