ESS111 : Programming 1 (C Programming)
LAB - 9

Due: 28 February, 2021 @ 11:59 pm

Part A (to be submitted)

Problem 1: Given a sequence of open and closing parenthesis, '(' and ')', write a (C) program to check whether the given input sequence is valid. Output `VALID` if the given sequence is valid or `INVALID` if it is not.
Note: Given sequence of parenthesis is valid if there is a matching closing parenthesis for every open parenthesis. At any point while reading the sequence, the number of '(' that has been already read is greater than or equal to the number of ')' that has been read. The valid sequence always starts with a opening parenthesis.

Sample Input 1:
)(()
Output 1:
INVALID

Sample Input 2:
((()))
Output 2:
VALID

Problem 2: A class has n students and each student has a roll number, name along with the marks obtained (out of 100) in three subjects S_1, S_2 and S_3. Let X_1, X_2 and X_3 denote the average of each of the subjects S_1, S_2 and S_3, respectively. Write a (C) program to display as output the student details such as roll number, name and grade obtained in each subject such that if $S_i \geq X_i, 1 \leq i \leq 3$, display the grade in the respective subject as `A`, else `B`.
Note: Define a `student structure` with attributes `roll number, name,` S_1, S_2, S_3. Read as input the number of students, n. For each student, read from the keyboard the name and the marks obtained in three subjects (marks must be non-negative). Roll number has to be incremented automatically for each student input received such that the roll number should be displayed as a 4-digit number with preceding zeros.

Sample Input 1:
3
Ram 55 67 89
Satya 39 77 48
Raj 41 40 79
Output 1:
0001 Ram A A A
0002 Satya B A B
0003 Raj B B A

Problem 3: Write a (C) program to multiply two 64-bit unsigned integers.
Note: All the inputs are unsigned values and read the inputs from the keyboard in the form of hexadecimal values. Display the output as well in the form of hexadecimal.

Sample Input 1:
1234567890123456 A2345B78901234FF
Output 1:
B88D7DB092CE8A1B7D2186342CF99AA

Problem 4: Write a (C) program to display distinct characters in the given input string.
Note: Retain the first occurrence of the character and remove the repetitions of the same character, if any.

Sample Input 1:
occurrence
Output 1:
ocuren

Sample Input 2:
event
Output 2:
evnt

Problem 5: Write a (C) program to create a *dynamic list* of integers. Unlike array, the size of the list is not fixed at the beginning and this list can grow dynamically. Memory needs to be allocated while inserting a new element into the list.
Input the elements to insert into list. Walk the list and print the integers.
Next, delete all the odd valued integers. Print the new list (remember it may end up being a NULL list).
Note: You need to create a `node structure` which has at least two attributes: one to store the integer value, another as a pointer to a neighbouring node that contains the next integer and so on. The last node points to a NULL pointer indicating end of the list.

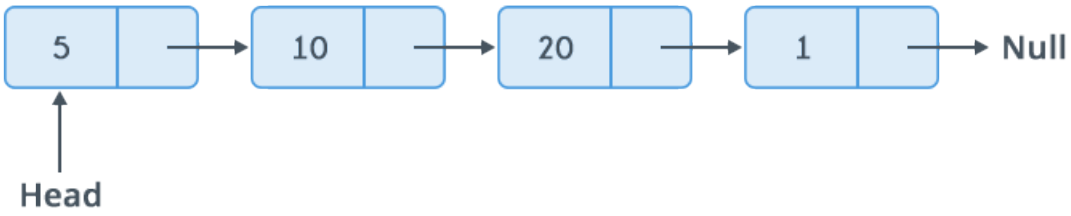

Figure 1: Dynamic list of 4 nodes

Sample Input 1:
3 6 5 9 13
Output 1:
3 6 5 9 13
6

Sample Input 2:
47 65 23 91
Output 2:
47 65 23 91
NULL

Problem 6 (Using function pointers): Define functions to implement the bitwise operators: `leftshift`, `rightshift`, `and(&)`, `or(|)`. Function implementing the bitwise operator takes two unsigned integer parameters and returns an unsigned integer as output. Consider the characters of 'L', 'R', 'A' and 'O' to indicate the operations of `leftshift`, `rightshift`, `and`, `or`, respectively. Read as input the character to indicate the bitwise operation along with two input numbers.
Write a (C) program to call the corresponding function using *function pointers*.
Note: The left shift and right shift operators should not be used for negative numbers. The shift value must be with in the range of 1 to 31 (both inclusive).

Sample Input 1:
255 10 L
Output 1:
261120

Sample Input 2:
121 15 O
Output 2:
127

Part B (need not be submitted)

1. Consider this C code to swap two integers and below five statements. Select the correct option below.

```
void swap(int *px, int *py)
{
   *px = *px - *py;
   *py = *px + *py;
   *px = *py - *px;
}
```

S1: will generate a compilation error
S2: may generate a segmentation fault at runtime depending on the arguments passed
S3: correctly implements the swap procedure for all input pointers referring to integers stored in memory locations accessible to the process
S4: implements the swap procedure correctly for some but not all valid input pointers
S5: may add or subtract integers and pointers.
(1) S1 (2) S2 and S3 (3) S2 and S4 (4) S2 and S5

2. What does the following fragment of C-program print?

```
char c[] = "GATEQUIZ";
char *p =c;
printf("%s", p + p[3] - p[1]) ;
```

(1) GATEQUIZ (2) EQUIZ (3) QUIZ (4) UIZ

3. Consider the following C function:

```
void swap (int a, int b)
{
   int temp;
   temp = a;
   a = b;
   b = temp;
}
```

In order to exchange the values of two variables x and y:
1) call swap (x, y)
2) call swap (&x, &y)
3) swap (x,y) cannot be used as it does not return any value
4) swap (x,y) cannot be used as the parameters are passed by value

4. What is printed by the following C program?

```
int f(int x, int *py, int **ppz)
{
  int y, z;
  **ppz += 1;
   z  = **ppz;
  *py += 2;
```

```
    y = *py;
    x += 3;
    return x + y + z;
}
void main()
{
    int c, *b, **a;
    c = 4;
    b = &c;
    a = &b;
    printf( "%d", f(c,b,a));
}
```

(1) 18 (2) 19 (3) 21 (4) 22

5. Choose the correct option to fill X and Y so that the program below prints an input string in reverse order. Assume that the input string is terminated by a newline character.

```
void reverse(void)
 {
  int c;
  if (X) reverse() ;
  Y
}
main()
{
  printf ("Enter Text \n") ;
  reverse();
  printf ("\n") ;
}
```

(1) X is (getchar() != '\n') and Y is getchar(c);
(2) X is (c = getchar()) != '\n') and Y is getchar(c);
(3) X is (c != '\n') and Y is putchar(c);
(4) X is ((c = getchar()) != '\n') and Y is putchar(c);

6. The most appropriate matching for the following pairs

X: `m=malloc(5); m= NULL;`	1: using dangling pointers
Y: `free(n); n->value=5;`	2: using uninitialized pointers
Z: `char *p; *p = 'a';`	3. lost memory

(1) X-1 Y-3 Z-2
(2) X-2 Y-1 Z-3
(3) X-3 Y-2 Z-1

(4) X-3 Y-1 Z-2

7. Consider the following C program segment:

```
char p[20];
char *s = "string";
int i, length = strlen(s);
for (i = 0; i < length; i++)
     p[i] = s[length - i];
printf("%s",p);
```

The output of the program is
1) gnirts 2) gnirt
3) string 4) no output is printed

8. Write a program to copy contents of one file to another. While doing so replace all lowercase characters to their equivalent uppercase characters.

9. Implement the problem given in Section A, **Problem 2** using *union*. Also, display the amount of memory required to store one student record. Observe the difference between the two approaches.

10. **(Usage of `argc, argv`):** Write a (C) program `expr` which evaluates an operation from command line inputs, where each operator or operand is a separate argument. For example:

```
expr 20 30 +
```

evaluates 20 + 30 and

```
expr 5 10 *
```

evaluates 5 * 10. Print out the result.

Note: Use command line arguments, namely, `argc and argv`, to read in the operators and operand. Make sure you check for the number of arguments before evaluating them.