Institute for Advancing Intelligence (IAI), TCG-CREST
Mid-Semesteral Examination
Ph.D Program Session: 2024–2025, Semester-I
Subject: Introduction to Programming and Data Structures

Date: 18. 10. 2024 | Full Marks: 40 | Time : 3 Hours

Instructions:

- Question 1 is compulsory. It must be submitted before starting the other questions. The maximum allocated time is 45 minutes. Answer any **two** from the remaining four questions.

.....

1. (a) Given a float value `f`, Alice and Bob tried to convert it to an integer and stored the results in `var_1` and `var_2`, respectively. They did it in two different ways, expecting the same converted values. Explain whether they will get the same result, and if not, why.

```c
float f = 1.0;
int var_1 = (int) f;
printf("%d\n", var_1);

int * temp_addr = (int *) &f;
int var_2 = * temp_addr;
printf("%d\n", var_2);
```

or

Someone has written the code below to display content of a file. Explain the correctness of the code. Explain the possible output.

```c
#include<stdio.h>
int main(){
    unsigned char c='x';
    FILE * f = fopen("file1.txt", "r"); //assume file exists
    while (c !=EOF){
        c = fgetc(f);
        printf("%c", c);
    }
    fclose(f);
    return 0;
}
```

[5]

(b) Write a C program that prints `0 1...14 15`, separated by spaces, in the terminal. But you are not allowed to use any semicolon in the whole code. Thus the output will be

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

or

Write a C program to print "your name" n times without using any loop (while, do-while, for, etc.) & goto statement. n should be taken as a user input from the terminal/console. Semicolon can be used if necessary.

Hint: see the input and output of the `printf` function.

[5]

2. A *binary tree* is a hierarchical data structure in which each node has at most two children, referred to as the left child and the right child. A *binary search tree (BST)* is a binary tree where each node's left subtree contains values smaller than the node, and the right subtree contains values larger than the node.

   (a) Create and initialize a BST.

   (b) Write a function to *insert* a value (positive integer) into a given BST.

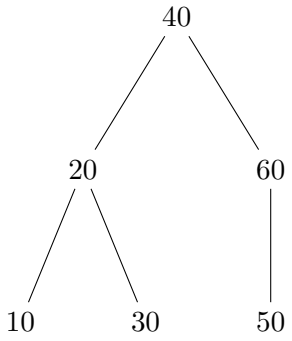
   (c) Write a function to *replace* a target value with a new value in a BST, if the target value is present. Note that the tree may no longer remain a BST after the replacement. [4]

   (d) Write a function to *check* if a given binary tree is a BST. [9]

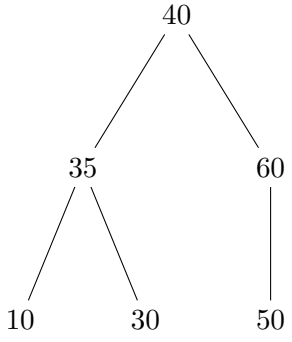
   (e) Write a menu to test the above three functions. [2]

   [4+9+2=15]

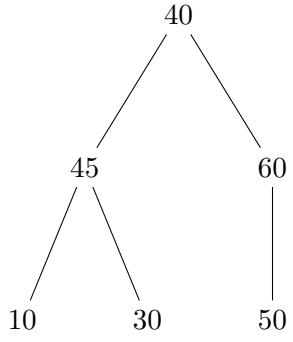
Example Input/output.



Initial binary tree.
It is a BST



After replacing 20 with 35
Binary tree is no longer a BST



After replacing 20 with 45
Binary tree is no longer a BST

3. You are given two files, say `matrix1.txt` and `matrix2.txt`, each containing a *monomial matrix*. In each file, the first row contains the dimension of the matrix, say n, followed by n rows, each containing n space-separated integers. *A monomial matrix is a square matrix in which each row and column has exactly one non-zero entry.*

   Write a matrix structure, `struct mono_matrix`, to store such a monomial matrix. Write a function to scan both matrices and store them in two `mono_matrix` structure variables. Then, multiply the matrices and display the resulting matrix in the terminal. In summary, the program should have the following functions:

   - `struct mono_matrix ScanMatrix(char *fileName)`: Scans the $n \times n$ matrix using **O(n) storage** and returns the matrix structure.
   - `struct mono_matrix MultMatrix(struct mono_matrix A, struct mono_matrix B)`: Multiplies two monomial matrices, A and B, and returns the resulting matrix.
   - `void display_matrix(struct mono_matrix A)`: Displays the matrix in the terminal.

   Restrictions: You **cannot** use more than **O(n) storage** to store a monomial matrix.

   Note: Function names, structure names, etc., are provided for better explanation only. You may choose your own names. [15]

   Example Input/output:
   Read input matrix A and B from files. Result $C = A \times B$ displayed in the terminal.

```
5
0  0  0  0  3
2  0  0  0  0
0  0  1  0  0
0 -1  0  0  0
0  0  0  4  0
```

`matrix1.txt`

```
5
0  0  0 -5  0
0  0  0  0  7
0  1  0  0  0
0  0 -2  0  0
9  0  0  0  0
```

`matrix2.txt`

```
5
27  0  0   0  0
 0  0  0 -10  0
 0  1  0   0  0
 0  0  0   0 -7
 0  0 -8   0  0
```

Output in the terminal

4. Create a doubly linked list where each node has two pointers—one pointing to the next node and the other pointing to the previous node. The next pointer of the last node and the previous pointer of the first node are set to NULL. Write the following functions:

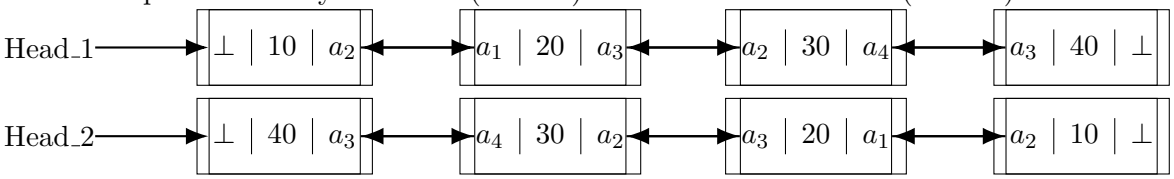
   (a) A function to add a node after the k-th node from the beginning. If there are fewer than k nodes, add the new node at the end.

   (b) A function to reverse a given doubly linked list *without changing the data part of any node.*

   (c) A function to display the doubly linked list.

   (d) Finally, write a menu in the main function to test the output of the functions. [5+5+2+3=15]

   • An example of a doubly inked list (Head_1) and its reversed version (Head_2).



   • a_1, a_2, a_3, and a_4 are the addresses of the 1st, 2nd, 3rd, and 4th nodes (in the list 1, represented by Head_1), respectively.

5. In the C programming language, **ellipsis** is a syntax that allows functions to accept an arbitrary number of arguments. Using ellipsis

   (a) Write a function that takes an arbitrary number of strings and concatenates them. Finally, return the concatenated string. Example:

Input	Output
`tcg crest iai`	`tcgcrestiai`
`C requires practice`	`Crequirespractice`
`10 20 30 40 50`	`1020304050`

   (b) Write a function that takes arbitrary number of strings and concatenate them but keep a separator symbols ('_', '*', '-', etc.) between the input strings. Finally, return the new string. Example:

Input	Output
`- tcg crest iai`	`tcg-crest-iai`
`_ C requires practice`	`C_requires_practice`
`* 10 20 30 40 50`	`10*20*30*40*50`

   (c) Write a main function and initiate 5 strings. Then call the above function passing first 1, 2 , 3, 4 and 5 strings respectively. Display the output strings on the terminal.

   Restriction: You can not use `<string.h>` library. [4+7+3=15]