CS23820 Assignment - C and C++

Document explaining what have I done and justifying my design.

Wojciech Sowinski
wos2
wos2@aber.ac.uk

At the very top of my membership program I included two standard libraries such as stdio.h and stdlib.h, and also string.h package. Then I defined a struct called node. Node in that case you can treat as Member struct but I decided to called it node as it behaves as node and it is a node of Binary Search Tree. It has 6 variables, 5 char arrays and 1 int to contain all information of member. It has also two nodes: left and right, that represent node (member) nodes children.

Just below I have a function of struct node type called **newNode**. It creates a new node of BST(binary search tree) and takes 6 parameters: family name, personal names, membership id, email address, class name of boat owned and name or number of boat. Within the function is defining an empty variable *temp of type struct node. Allocating a memory for the node using malloc and then it assigning parameters to variables of temp node using **strcpy** in case of char arrays variables or operator assigning in case of int. Assigning family_name from parameters to temp→family_name and so on. Then as it is a new node function it has no children, so temp→left and temp→right assigning NULL. At the end, function returns temp.

Next function you can see is **insert** function of type struct node*. It is responsible for inserting nodes into BST. It takes seven parameters, first is a variable *root of type struct node, rest of parameters is the same as in case of newNode: family name and so on… First, the function checks whether root is NULL or not. If root == NULL, function returns other function newNode passing the rest 6 parameters to newNode function and creates the root. Otherwise, if root already exist, insert function compares key (family name) with root→family name. Because both variables are type of char array I had to use strcmp function to compare these two. If parameter family name is greater (based on strcmp) than root→family_name variable – it assigning to root's right child all the parameters, adding by that a new node (member) into BST. Then it returns the root. However if parameter family name is less than root→family_name – it adds a member to root's left child and returns the root. End of function.

Function **inorder**() is ordinary inorder traversal function in case of BSTs and takes just one parameter of type struct node. In my program it is responsible to print all members in order of their family names. It based on recursion. First checks if root is not NULL, if it is – performs recursion of inorder where it takes left child of root as a parameter, printouts root variables and performs inorder functions but this time takes right child as a parameter.

Void function **saveIntoFile** takes struct node *parameter and file's name, creates a FILE pointer and then open the file. Checks whether file exists, if it doesn't – exits(1), otherwise works same as inorder() function. It based on recursion. Performs saveIntoFile which takes left child as one of parameters, then writes into file using **fprintf** all root's data and then performs again saveIntoFile which takes this time right child as one of parameters. Closes the file.

Other important function is function **search** of type struct node that takes two parameters – root and char array. First it checks if root is NULL – if it is returns

NULL. If not, it checks using strcmp through BST using keys to compare with family name from parameters and checks if they are the same. If they are – printout the root variables and then return the root. However if parameter family name is based on strcmp greater than root→family_name it returns function search that takes new parameters of root's right child but the same family name (recursion here). Else it returns recursive function search but takes left child of root as one of parameters. It runs until it finds search key or there's no such a key in BST.

SearchClass functions is responsible for looking through BST for members that owe boat of a given class. It takes two parameters – root and class name. is also based on recursion. It searches every single node. First, if root is not NULL it checks if current looked root has the same boat class name, if so – printout current looked root details, then performs recursion in form of searchClass twice. First takes left child as one of parameters and just after takes a right child.

Function **searchID** works basically in the same way as searchClass I decsribed -above but it takes different parameter – id of type int. And compares it to root→membership_id. If they are the same – printouts current looked root, else performs searchID twice with different children of root.

In the **int main()** first that I do is creating a root of type struct node and assigning it to NULL. Then I define 6 variables related to format of members' information. Family name, personal names and so on, using the same type of variables as at the top of main.c, to use it for reading the file or look for member in BST using one of them as parameter. I created pointer to file, char array called filename that I assign to user input where is asked to provide file name to open and variable int number, which I use to read file till the end of the file. Program takes user input to open a file specified by that with mode "r+". Exits if file does not exists. If exists performs scanning file until end of file with appropriate format (e.g. " %[^\n]s" to avoid unnecessary spaces) and uses insert() function that takes read variables as parameters if root is not NULL. If root is NULL then assigns root to return of insert() that takes read variables as parameters. File is closed.

Below is menu to printout and switch expression to run until user decides to exit the program. Each case is responsible to run the specified function. I do not think I need to explain that except option which user decides to save BST into a file. Because function is recursive I couldn't overwrite program using just mode "w+" while fopen as it will just write first member. So I decided to remove .txt file and then create a new one with the same name and write BST into new created new using "a+" mode of fopen. I understand that is not the way you wanted me to do it – it's not, but at least it works :)

I have decided to not attach the other program as I couldn't make it done or even do a significant progress. However I spend many, many, many hours on this assignment and I'm very satisfied with the result as at first I did not give me many chances to finish even the first program – but hey, I learnt a lot of C and about pointers so I'm satisfied. This was interesting experience and I had fun, even though I was initially scared of C. Thank you for your attention.