CMPT225 Assignment 3 Report

Concept:

For this project, I have a BinaryHeap class that stores the priorities in minimum heap array and a AvlTree class that stores project numbers. The binary heap is implemented as an array and each array element is a heap node that contains priority and a pointer to the AVLNode in AVLtree. The AvlNode is declared in a separate file called AvlNode.h so that later on the PQ class have access to it. The AVLNode contains project number, heap index, pointer to left, pointer to right and the height.

The third class is the PQ class has an AvlTree object called taskTree and a BinaryHeap object called priorityList. The tastTree stores the project numbers in a way that we can quickly traverse the tree and find the project number in O(logn). And the reason chose to store priority in min heap is that we can efficiently find the top priority at index 1.

<u>Below are the functions added or modified:</u>

PQ.h

- **Private variables:** Object of AvlTree class and object of BinaryHeap class
    - These two objects communicate using the pointers and array index.

AvlTree.h

- **void set_index(const Comparable & x, const int & new_index) const**
    - This function change the heapIndex of AvlNode that has x as project number to new_index.
- **AvlNode<Comparable>* get_avlNodePtr_from_prj(const Comparable & x) const**
    - This function returns the pointer to the node that has x as project number
- **AvlNode<Comparable>* get_avlNodePtr_from_ind(const int & ind) const**
    - This function returns the pointer to the node that has ind as index
- **void insert(const Comparable & x,const int & y)**
    - Modified the insert function to take two arguments and store them at prj and heapIndex in the AvlNode.
- **void displayLinks(AvlNode<Comparable> *t, int depth, ostream & out)const**
    - This prints out the internal structure of the AVL project tree. It shows the project number with its index of where its priority is stored. And it shows the address of the nodes so later can cross check with heap array's pointer.

BinaryHeap.h

- **int get_currentSize() const**
    - return the size of current priority list which is used in the size() of PQ class
- **int get_index(int x) const**
    - returns the index of the node that contains x for priority
- **Comparable get_ptr2AVL(int ind) const**
    - returns the pointer to AVL node at array element of index 'ind'.
- **void set_ptr_by_priority(const int prio, const Comparable1 new_ptr**)

    - Set the ptr of array element that has ‘prio’ as priority to new_ptr
- **void set_ptr_by_ind(const int ind, const Comparable1 new_ptr)**
    - At the element of index ‘ind’, set the ptr to new_ptr
- **bool priorityRepeated(int p)**
    - Checks if the priority given exist in array. This helps prevent inserting task with non-unique priority.
- **void update_priority(const int & arr_index, int new_prio)**
    - This function set the priority of the element at index arr_index to new_prio
- **void insert(const int & x, const Comparable1 &y)**
    - The insert function is modified so it takes two argument and made them into a heapNode then insert to the array.
- **void displayArray()**
    - This display the priorityList by showing its priority number and the address the AVLNode pointer points to at each index. The address can be verified by matching it with the AVLnode’s address in the taskTree.

AvlNode.h

- **AvlNode**
    - Implemented as a struct in an individual header file. It was pulled out of the AvlTree class so that the PQ class can take AvlNode pointer for its template type and initialized an array with AvlNode pointers in its element.