Cüneyt EREM, Sec 1, 21202398, homework3

1)

1. I inserted all values one by one according to AVL tree rules

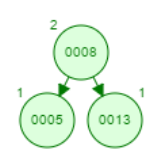
Insert 13



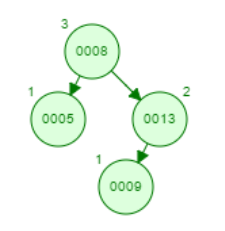
8



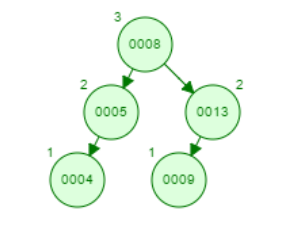
5



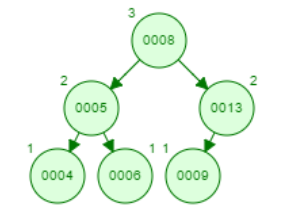
9



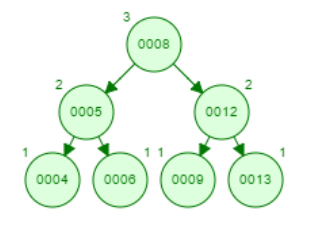
4



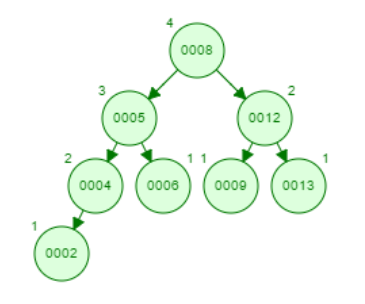
6



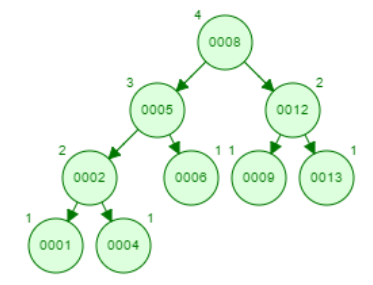
12



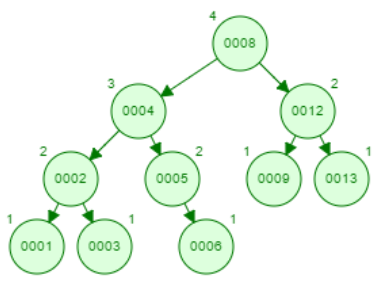
2



1



3



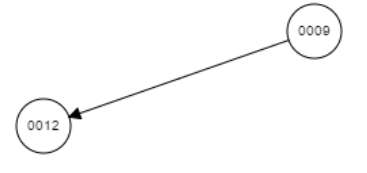
b)

Inserting min heap one by one

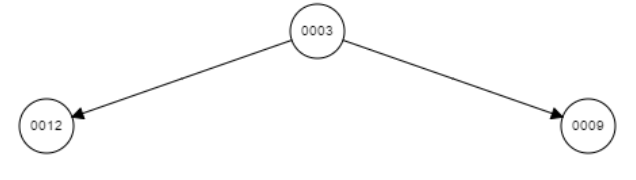
Insert 12



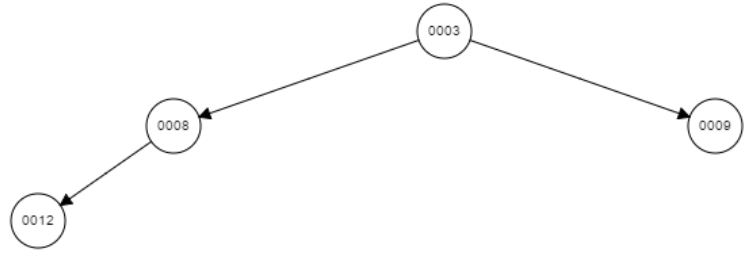
9



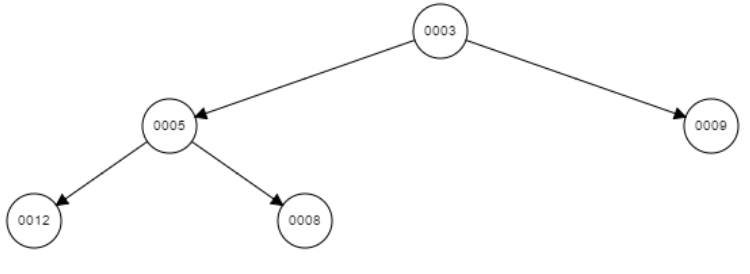
3



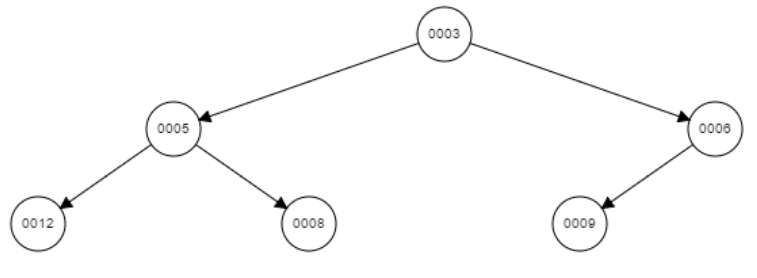
8



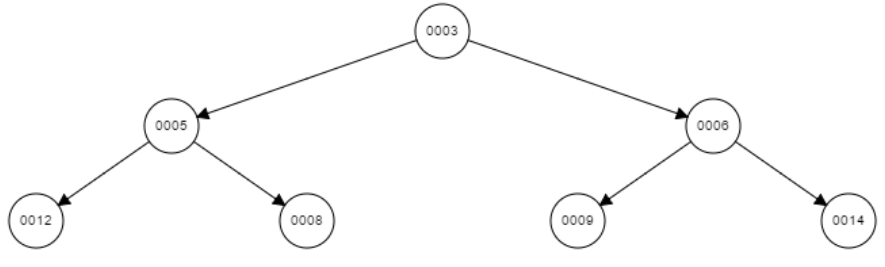
5



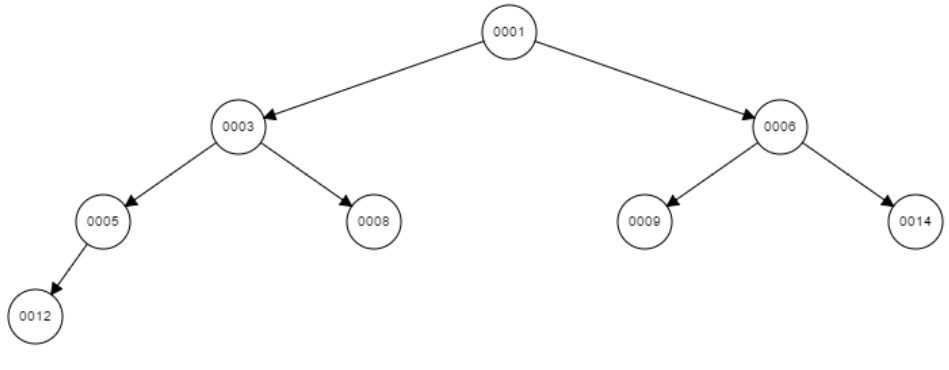
6



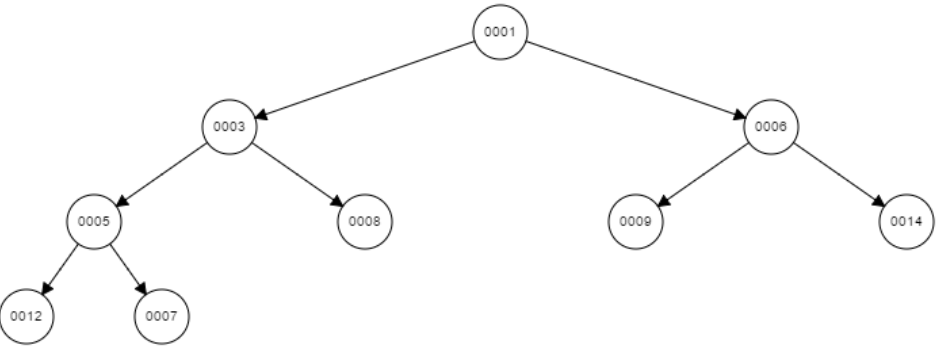
14



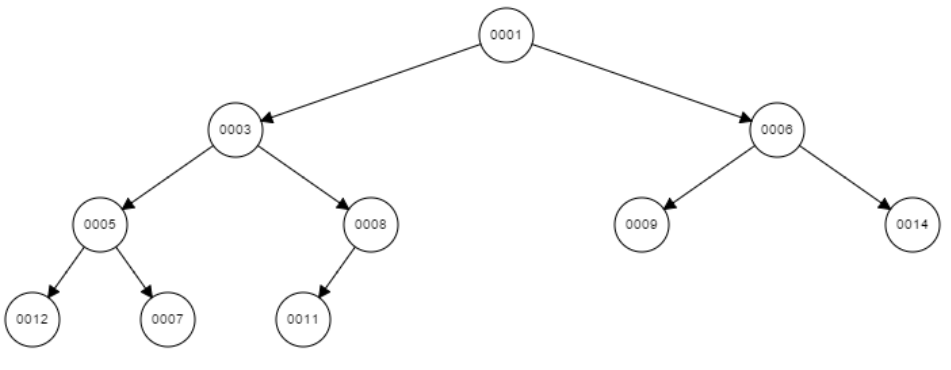
1



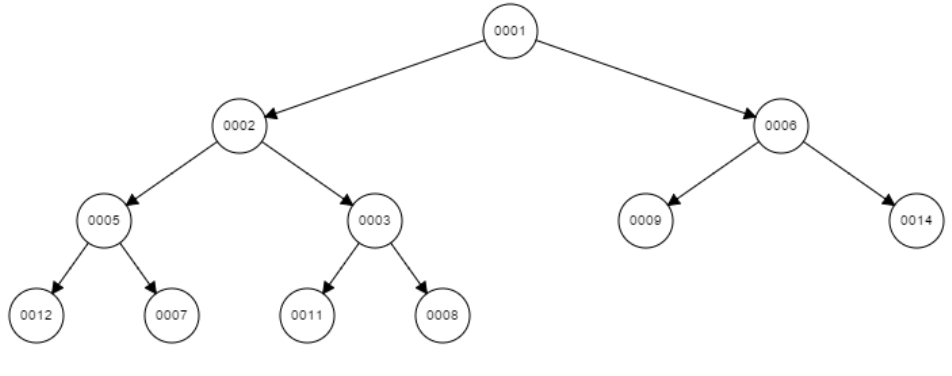
7



11

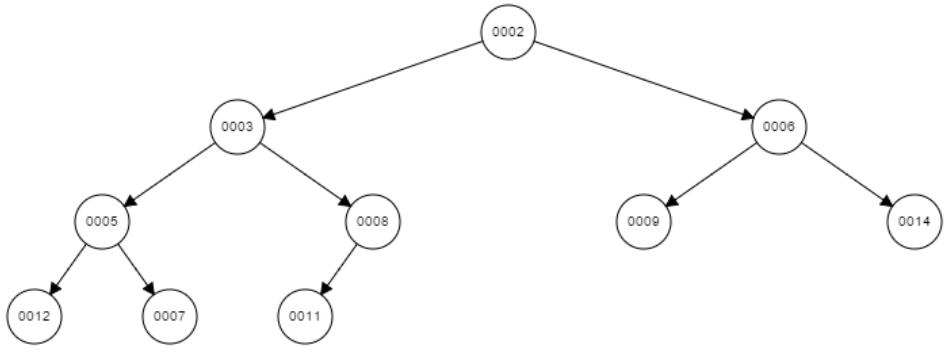


2

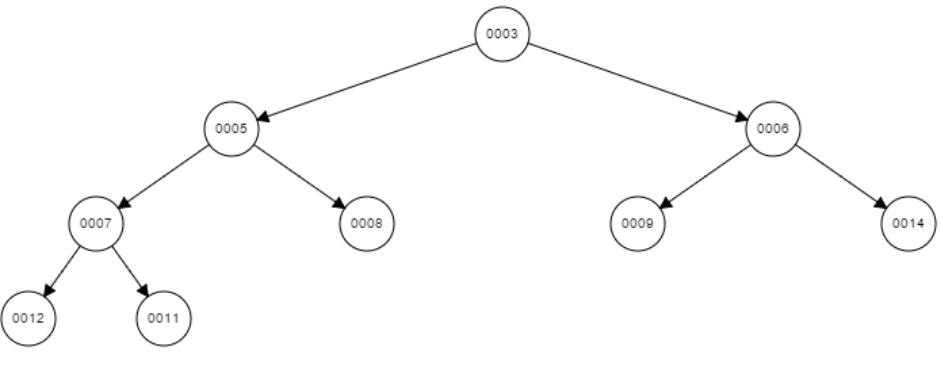


Delete Op;

deleteMin1

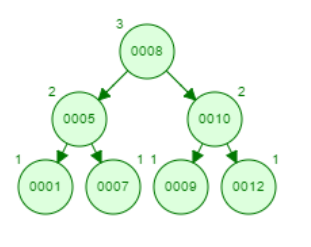


deleteMin2



C)

For example this is our heap tree;



Pre: 8, 5, 1, 7, 10, 9, 12

Ino: 1, 5, 7, 8, 9, 10, 12

Post: 1, 7, 5, 9, 12, 10, 8

As you see, any of pre/in/post order do not match the sorted order because for the heap tree, important thing is parent has to have big value than his children. And, comparison of children’s value is not important. Also, because of height condition, we cannot regulate the order for the pre, in and post. All of orders have their rules, and in AVL, we cannot obtain sorted order because of AVL situation.

d) To find min num of nodes in AVL, expression is;

S(h) = s(h-1) + s(h-2) +1

SO, for h = 15;

S(1) = 2;

S(2) = 4;

S(15) = s(14) + s(13) +1

S(14) = s(13) + s(12) +1

……………..

S(5) = s(4) + s(3) +1= 20

S(4) = s(3) + s(2) +1= 12

s(3) = 2

* S(15) = 2583

(head is not included in height)

e)

to determine minheap from the binary tree, we have to traversal children to parents and compare the parent to children values;

bool determine(Tree root) {

if(root < leftchild == NULL && root < rightchild == NULL)

return true;

determine(root->leftchild);

determine(root->rightchild);

if(root < root->leftchild && root < root->rightchild)

return determine(root->leftchild) && determine(root->rightchild);

else

return false;

}

2)

(-codes are in the files, all 5 outputs are also in the file, program works correctly)

Report;

//heap.h class, int size and comparison will be used, and we need other methods.

//heap.cpp class, constructor that has initial values

//destructor deletes the array not to cause memory leak

//for inser method, if size is not enough, then it will increase the size of array, and newitem will be inserted last index, after the insertion, we rebuild the heap and increase the size

//for buildheap method, counter for using buildheap everytime, we have a array structure that to find left and right children indexes, if childrens are less than size, than we sont have to check other array elements after the last elemenets we have to check, and if-else comparison is to find item's exact place, then rebuld the heap, then one more if c. to find parent index and go to upper location and swap

//swap method swaps the value

//maximum method it will returns the head of the tree

//pop max method, swap the last element with head element and decrease the size of element, after then rebuild heap.

//and getsize, isempty and getcomparisons methos that return their values.

//heapsort.cpp class, if c. for number of parameters that we will write to cmd for heapsort.exe, input.exe and output.exe

//read input and outputs

//read lines as integer from the input files

//after end of the input, pop the maximum element and write the values as sorted

//finally, prints the number of comparisons

For the 5 different inputs;

Our theoretical worst case formula is 2 \* n \* logn (2 is for insert and delete, n number, and logn notation)

Word theoretical actual

Input1: 1000 20 000 10589

Input2: 2000 44000 23260

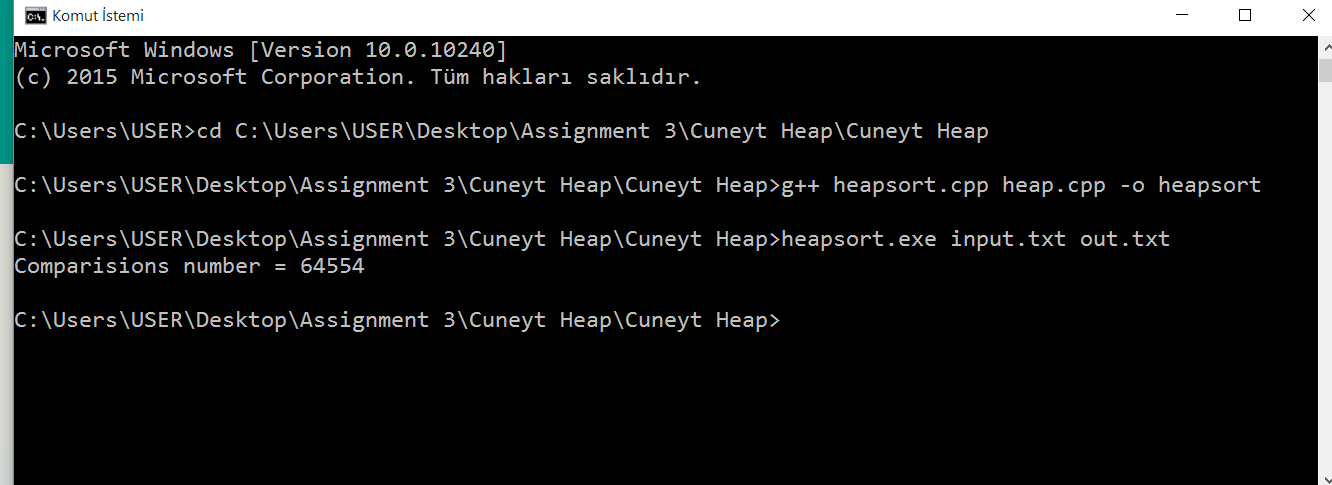
Input3: 3000 69300 36536

Input4: 4000 95680 50517

Input5: 5000 123000 64554

As we can see, actual number are less than theoretical number because, theoretical is for the worst case, so these result are about average case which we expected.

(program works;



)