HW 4 Report

Part1 : ***Size()***: I returned cur_size to get the number of items in the priority queue.
PercolateUp(): We check if there's a parent of a given position, and compare the Node with each other, if the current Node with value is smaller than that of it's parent node, it swap the two Nodes and keep going until the while loop breaks.
PercolateDown(): When the right child has a child, we define a child and equal to the left child by default. And if the Node has a right child node and it is smaller than the left child, then change the value of child into right child. After that if the rightNode value is smaller than the current Node, swap the two nodes with each other and after that set the current Node to child node which we declared in the while loop.
Top(): Basically return the root node which is at the top.
Push(): We use pushback to push the item and std::move to transfer the content of the object item to the vector items. Add 1 to the cur_size and procolate up since it needs to put the largest/smallest value on the top according to the characteristics of the comparator.
Pop(): The last item now is at the root()/Top, reduce the cur_size by 1, and remove the last item by using pop_back. Then procolate down the root item to let it follow the character of the priority queue.
CompareNodes(): We are comparing the two items' value within the vector, and the cmp is introduced to handle it, since it has the same type as typename C, it will be eligible when comparing some special cases. For our CompareNode(), we return true, while the cmp statement returns true, and returns false otherwise.

Part2

because getchar will receive 8 bits from the buffer, the buffer will flash automatically. the unread bits will remain in the buffer, until we flash the buffer. Getint is the same as getchar, but receives 32 bits. Put method is just repeating the putbit, its destructor will be called before closing the file, and flush the remaining bits in the buffer to the target file. for test case, I test all getbit, getchar,getint, putbit, putchar,putint individually, and make few combination between those.

Part3

compress: we first read all contents into a vector map, which can be easily pushed into a priority Q. The Q is defined as a type of huffmannode class, then we build the tree with the function that Q provided to us. the code table will be generated by the Q, also write bits into the output file at the same time. The root of Q will tell us how many characters in total. writing the contents into the output file by the output table.
decompress: based on the Huffman tree values, we can generate the code table. We generated a tree for the code table, because all 1 are a leaf, so the recursion function will stop, after all the left nodes are full. It indicated that it is the end of Huffman tree and the star of the Number of encoded characters. To decode sequence characters, we use a similar idea as building a tree, but use for, if loop to implement it. for example, we have 1111001, and have a code table of a=11,b=0,c=01. create a root node for the tree, turn left or right by the input bit, if we find a leaf node, return the data where the pointer is currently pointing at.