Student Name : Rakyan Satrya Adhikara

Student ID : 219548135

Task 6.1 (Search and Stack)

Question 1:

Use Merge Sort to sort the array in steps (worst case). Then go through the array and for each element use binary search to determine if is in the array.

The Merge Sort Pseudocode:

MergeSort(Array)  
{  
 if (array.length <= 1)  
 {  
 return array  
 }   
 left = new array  
 right = new array   
 mid = left + right/2  
 MergeSort(left)  
 MergeSort(right)  
 return merge(left, right)  
}

The Binary Search Pseudocode:

BinarySearch(Array, value)  
{  
 while(start <= end)  
 {  
 mid = (start + end)/2  
 if(Array[start] + Array[end] < Value)  
 {  
 start = mid + 1  
 }  
 if(Array[start] + Array[end] > Value)  
 {  
 end = mid - 1  
 }  
 if(Array[start] + Array[end] == Value)  
 {  
 return mid  
 }  
 }  
 return not\_found  
}

Question 2:

We use two stacks. One for an actual stack itself, and one for a stack that only consists of minimum element.

Properties in stack:

Stack[] // The actual stack  
MinStack[] // The stack that only contains minimum  
Count // Length of the stack  
CountMin // Lenth of the MinStack  
Min // Current minimum element in the stack

For *Insert* operation:

StackPush(Value)  
{  
 if (stack is full)  
 {  
 throw new StackOverflowException()  
 }  
 count++  
 if (Value < Min)  
 {  
 Min = Value  
 MinStack.Push()  
 }  
 stack[count] = Value  
}

For *Delete* operation:

StackPop()  
{  
 if (stack is empty)  
 {  
 throw new InvalidOperationException()  
 }  
 result = Stack[Count]  
 count—  
 if (result == MInStack[CountMin])  
 {  
 MinStack.pop()  
 CountMin--  
 Min = MinStack[CountMin]  
 }  
 return result  
}

For *GetMin* operation:

GetMin()  
{  
 return min  
}