Assignment 1 Rapor

```
90  void insertionSort(Array& myArrayAdress, int size) {
91      cout << "insertionSort function is working..." << endl;
92      int j, temp;
93
94      for (int i = 0; i < size; i++) {
95          j = i;
96
97          while (j > 0 && myArrayAdress.myItem(j) < myArrayAdress.myItem(j - 1)) {
98              temp = myArrayAdress.myItem(j);
99              myArrayAdress.add(myArrayAdress.myItem(j - 1), j);
100             myArrayAdress.add(temp, j - 1);
101             j--;
102         }
103     }
104     cout << "insertionSort function is done." << endl;
105 }
106
```

Yukarıda gösterilen InsertionSort fonksiyonunda her satırın T(n) değeri aşağıdaki tablodaki gibidir.

Satır	T(n)
91	1
92	1
94	n
95	n
97	S
98	S-(n)
99	S-(n)
100	S-(n)
101	S-(n)
104	1

Yukarıdaki tablo incelendiğinde sıralama işlemi yapılacak array'in içinde ki değerler sıralı halde ise; S değeri n değerine eşit olacak ve bu fonksiyonun en iyi değeri O(n) olacak. Eğer değerler tersten sıralanmış halde ise; S değeri n(n+1)/2 değerine eşit olacak ve en kötü değeri olan O(n^2) değerini alacak.

Aşağıda yer alan resimde; mergeSort fonksiyonu var olan n elemanlı array'yi n/2 elemanlı 2 parçaya ayırıyor. Daha sonra her n/2 elemanlı parçayı (n/2)/2 elemanlı 2 parçaya ayırıyor ve bu işlemleri 1 eleman elde edene kadar yapmaya devam ediyor. Daha sonra bu 1 elamanlı arrayleri merge fonksiyonuna gönderiyor. 1 elemanlı arrayleın sıralanması O(1). Bu işlem n tane elemana uygulanıyor. Parçalama işlemi ise her seferinde ortadan böldüğü için O(lgn) değerine sahip. Bunun sonucunda O(nlgn) değerine sahip oluyor.

```
107 void mergeSort(Array& myArrayAdress, int low, int high) {
108     //cout << "mergeSort function is working..." << endl;
109     int mid;
110
111     if (low < high) {
112         mid = (low + high) / 2;
113         mergeSort(myArrayAdress, low, mid);
114         mergeSort(myArrayAdress, mid + 1, high);
115         merge(myArrayAdress, low, mid, high);
116     }
117     //cout << "mergeSort function is done." << endl;
118 }
119
120 void merge(Array& myArrayAdress, int low, int mid, int high) {
121     //cout << "merge function is working..." << endl;
122     int counter1, counter2, counter3;
123     Array myTempArray(high + 1);
124
125     counter1 = low;
126     counter2 = low;
127     counter3 = mid + 1;
128
129     while (counter1 <= mid && counter3 <= high) {
130         if (myArrayAdress.myItem(counter1) < myArrayAdress.myItem(counter3)) {
131             myTempArray.add(myArrayAdress.myItem(counter1), counter2);
132             counter2++;
133             counter1++;
134         }
135         else {
136             myTempArray.add(myArrayAdress.myItem(counter3), counter2);
137             counter2++;
138             counter3++;
139         }
140     }
141
142     while (counter1 <= mid) {
143         myTempArray.add(myArrayAdress.myItem(counter1), counter2);
144         counter2++;
145         counter1++;
146     }
147
148     while (counter3 <= high) {
149         myTempArray.add(myArrayAdress.myItem(counter3), counter2);
150         counter2++;
151         counter3++;
152     }
153
154     for (counter1 = low; counter1 < counter2; counter1++) {
155         myArrayAdress.add(myTempArray.myItem(counter1), counter1);
156     }
157
158     //myTempArray.~Array();
159
160     //cout << "merge function is done." << endl;
161 }
```

Eleman sayısı	Insertion Sort	Merge Sort
1000	0,003679	0,000393
10000	0,344815	0,004946
100000	36,112465	0,044498
1000000	3.586,8145760	37,968274