Karadeniz Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü
2016-2017 Bahar Dönemi, Prog. Giriş Arasınavı, 24 Nisan 2017, 10:00, Süre: 2 saat

1. Sayılardan oluşan bir kümenin birbiri ile kesişmeyen iki alt kümeye bölünmesi istenmektedir.

   a) Altkümelerdeki sayıların toplamları arasındaki farkı maksimum yapacak bir algoritma öneriniz. Önereceğiniz algoritmayı "Big-O" notasyonunu kullanarak sınıflandırınız. (**15p**)

   Alt kümelerden biri sayıların en küçüğünü, diğeri ise geri kalan sayıları içerir. Sayıların en küçüğünü bulmak için en fazla n adet karşılaştırma gerekir (n: Orjinal kümenin eleman sayısı). Dolayısıyla en kötü durum performansı O(n) 'dir.

   b) Altkümelerdeki sayıların toplamları arasındaki farkı minimum yapacak bir algoritma öneriniz. Önereceğiniz algoritmayı "Big-O" notasyonunu kullanarak sınıflandırınız. (**15p**)

   En iyi olmayan fakat en iyiye yakın sonuçlar veren çözümler mevcuttur. Örnek: Orjinal küme büyükten küçüğe sıralanır (s_1 > s_2 > … > s_n). Alt kümelerden biri s_1 'i, diğeri s_2 'yi içerir. Geri kalan sayıların her birinin hangi altkümeye ait olacağının kararı hangi kümeye ait olması durumunda toplamlar farkının minimum olacağına göre verilir. Orjinal kümeyi sıralamak için en fazla n^2 kıyaslama gerekir (Bubble Sort kullanılması durumunda). Her bir sayının ait olduğu kümenin

belirlenmesi için en fazla n adet kıyaslama gerekir. Dolayısıyla en kötü durum performansı O(n^2 + n) = O(n^2) 'dir.

2. Tablo I 'de verilen prosedür `myFunction(4)` şeklinde çağrıldığında hangi ekran çıktısını üretir ? (**20p**)

Tablo I

```
procedure myFunction (N)
if (N > 0)
      then (N 'nin değerini ekrana yazdır ve
           myFunction 'ı (N-2) argümanı ile
           çağır)

(N+1) 'in değerini ekrana yazdır
```

CEVAP: 4 2 1 3 5

3. Lütfen aşağıda verilen C kodunun çıktısını operatör önceliklerini göz önünde bulundurarak belirtiniz. (**20p**).

Tablo II

```
int main()
{
    int a = 1; int b = -3; int c = 8; int d = 3;
    int e = a + (b ++) + c * b - d;
    if (e > b - c * (--d))  printf("Hello world %d!\n", e);
    else  printf("Goodbye world %d!\n", e);
    return 0;
}
```

CEVAP : Goodbye world -21!

4. Bir programcı verilen **insertion sort** algoritmasını Tablo III'te verilen şekilde kodlamıştır (**30p**)
   a) Kodda bulunan **iki mantıksal hatayı** bulunuz ve düzeltiniz.
   b) Düzeltilmiş algoritmanızla sizden herhangi bir dizinin **belli bir bölgesini** sıralamanız istenseydi bu fonksiyonunuzu nasıl değiştirirdiniz?
   c) Verilen fonksiyon integer bir diziye artan şekilde sıralamayı hedeflemektedir, bu fonksiyonu **bir karakter dizisini istenilen yönde** sıralayacak şekilde nasıl düzenlersiniz?

Tablo III

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 15

//prototype
void insertion_sort(int sorted_array[], int sorting_size);

int main()
{
  int array_to_be_sorted[SIZE] = {3, 5, 8, 2, 1, 6, 4, 7, 12,
  10, 9, 15, 11, 14, 13 };
  insertion_sort(array_to_be_sorted, SIZE);
  return 0;
}

void insertion_sort(int sorted_array[], int sorting_size){
  int counter = 1;
  int i, temp;
  int pivot = sorted_array[1];

  while(counter < sorting_size) {
    for(i = counter - 1; i >= 0; i--){
      if(pivot < sorted_array[i]) {
        temp = sorted_array[i];
        sorted_array[i] = pivot;
        sorted_array[i + 1] = temp;
      }
    }
    pivot = sorted_array[counter ++];
  }
}
```

a)

```
void insertion_sort(int sorted_array[], int sorting_size){
    int counter = 1;
    int i, temp;
    int pivot = sorted_array[1];

   while(counter < sorting_size) {
     for(i = counter - 1; i >= 0; i--){
       if(pivot < sorted_array[i]) {
         temp = sorted_array[i];
         sorted_array[i] = pivot;
         sorted_array[i + 1] = temp;
       } else break;
     }
     pivot = sorted_array[ ++ counter];
   }
 }
```

b)

```
void insertion_sort(int sorted_array[], int start_point, int sorting_size){
  int counter = start_point + 1;
  int i, temp;
  int pivot = sorted_array[start_point + 1];
  while(counter < sorting_size) {
    for(i = counter - 1; i >= start_point; i--){
      if(pivot < sorted_array[i]) {
        temp = sorted_array[i];
        sorted_array[i] = pivot;
        sorted_array[i + 1] = temp;
      } else break;
    }
    pivot = sorted_array[ ++ counter];
  }
}
```

c)

```
void insertion_sort(char sorted_array[], int start_point, int
sorting_size, int ascending){
  int counter = start_point + 1;
  int i, temp;
  int pivot = sorted_array[start_point + 1];

  int sign = ascending?1:-1;
  while(counter < sorting_size) {
    for(i = counter - 1; i >= start_point; i--){
      if(pivot * sign < sorted_array[i] * sign) {
        temp = sorted_array[i];
        sorted_array[i] = pivot;
        sorted_array[i + 1] = temp;
      } else break;
    }
    pivot = sorted_array[ ++ counter];
  }
}
```

	Soru 1	Soru 2	Soru 3	Soru 4
PÇ	1,2,3	1,2,3	1,2,3	1,2,3