**Not: Soruların cevapları ilgili boşluklara yazılacaktır (word de type edilecek, elle yazılmayacak). Sonra dosyanız pdf dosyasına dönüştürerek UBYS sistemine yüklenmelidir.**

**2)** Hash veri yapısı ile ilgili olarak aşağıdaki tanımlamalar verilmektedir. **(30 puan)**

**struct cell {**

**char \*anahtar;**

**struct cell \*next;**

**};**

**struct table\_node{**

**int count;**

**struct cell \*header;**

**};**

**struct hash\_tablosu{**

**struct table\_node \*tablo\_basi;**

**int tablo\_uzunlugu;**

**int multiplier;**

**};**

**unsigned int hash(char \*s, int multiplier, int table\_size){**

**int i=0;**

**unsigned int value =0;**

**while(s[i] != '\0'){**

**value=(s[i]+multiplier\*value) % table\_size;**

**i++;**

**}**

**return value;**

**}**

Buna göre, düğüm tanımlaması struct cell olan bağlı bir listedeki düğümleri hash tablosuna yerleştiren ve tanımlaması aşağıda verilen fonksiyonu yazınız. Not: Bu fonksiyonun kodlanmasında malloc fonksiyonu çağrımı yapmamanız gerekir. Sadece yapmanız gereken şey, (bağlı liste ile hash tablosundaki listelerin düğüm tanımlaması aynı olduğu için) bağlı listenin düğümlerini aşlangıçta boş olan hash tablosuna yerleştirmenizdir. Fonksiyondan geriye dönüldüğünde liste, boş liste şeklini almalıdır.

**3)** Derste heap veri yapısının dizilerle gerçekleştirimini görmüştük. Heap veri yapısının aşağıda tanımlanan değiştirilmiş ikili ağaç düğümleriyle gerçekleştirildiğini kabul edelim. dugum veri yapısındaki **parent** değişkeni ilgili düğümün parentine ilişkin işaretcidir (bir düğümden parentine erişimi sağlamak amacıyla eklenmiştir). heap veri yapısındaki **en\_son\_eklenen** değişkeni ise veri yapısına en son eklenen düğüme ait işaretiyi göstermektedir. Buna göre tanımlaması aşağıda verilen ve heap’e ekleme yapan fonksiyonu yazınız? **(30 puan)**

struct dugum {

int anahtar;

struct dugum \*parent, \*sol\_link, \*sag\_link;

};

struct heap{

struct dugum \*kok;

struct dugum \*en\_son\_eklenen;

}

**void heap\_insert(struct heap \*heap,int key){**

**struct dugum \*dugum = NULL;**

**struct dugum \*d = NULL;**

**struct dugum \*geri = NULL;**

**d = heap->kok;**

**for (;d != NULL;)**

**{**

**geri = d;**

**if (key < d->key){**

**d = d->sol\_link;}**

**else if (key > d->key)**

**d = d->sag\_link;**

**else**

**return;**

**}**

**struct dugum \*d1 = (struct dugum \*)malloc(sizeof(struct dugum));**

**dugum = d1->key = key;**

**d->sol\_link = d->sag\_link = NULL;**

**if (heap->kok == NULL)**

**{**

**heap->kok = dugum;**

**return;**

**}**

**if (key < geri->key)**

**geri->sol\_link = dugum;**

**else**

**geri->sag\_link = dugum;**

**}**