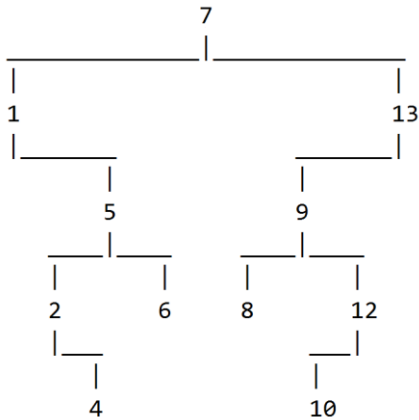
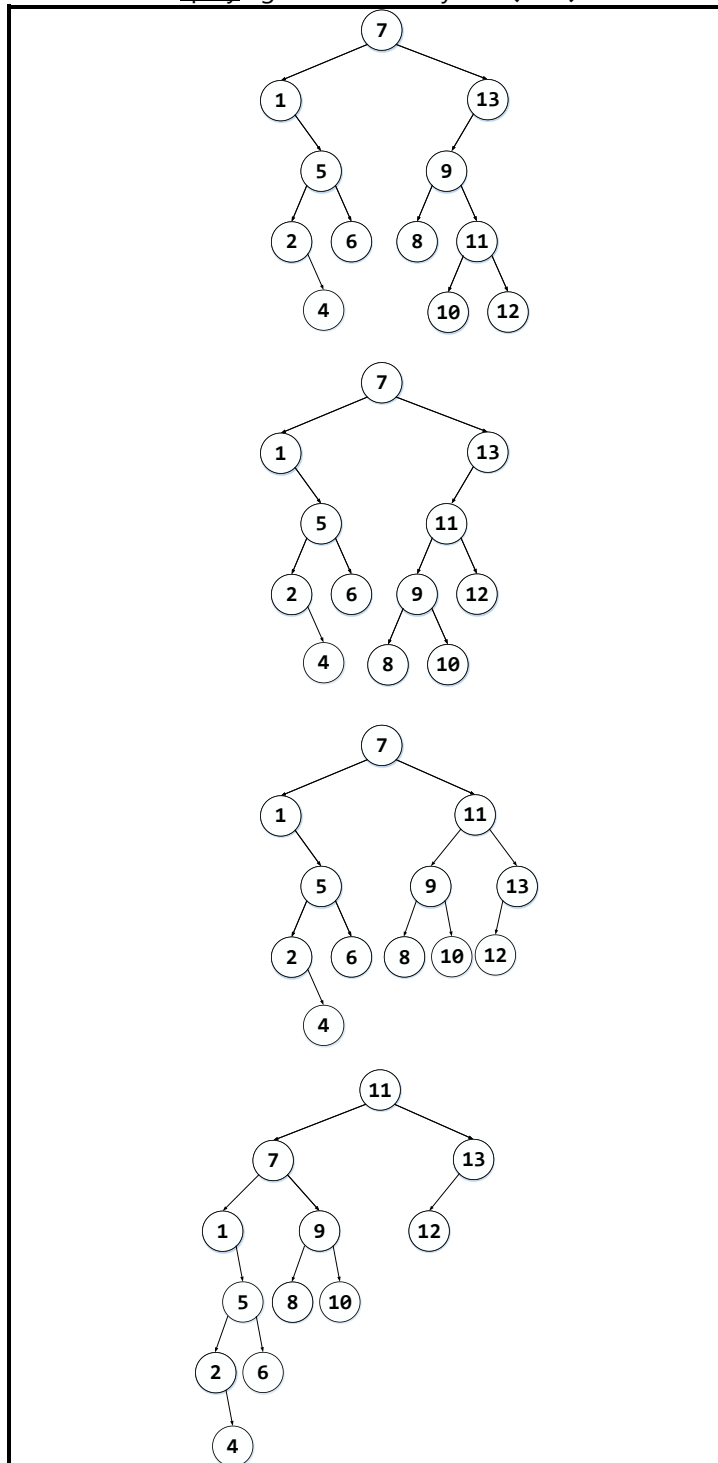




A GRUBU CEVAPLAR



1. Yukarıdaki splay ağacına 11'i ekleyiniz. (30P)



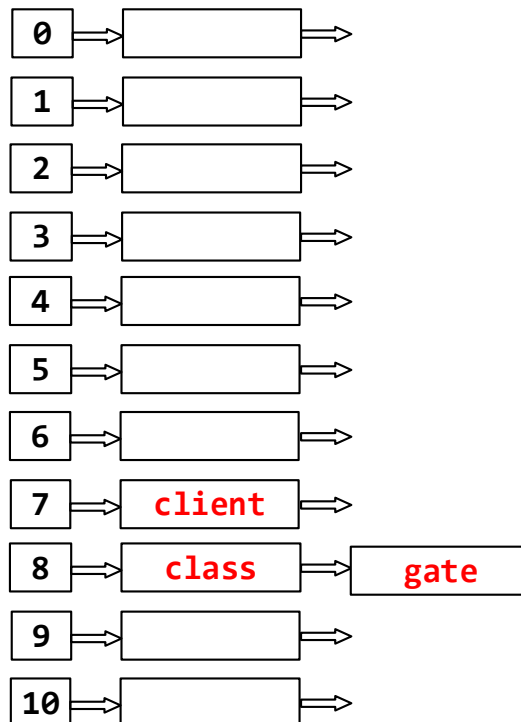
```
int Hash (char* key)
{
    int sum = 0;
    for (int j=0; j<4; j += 2)
        sum += 3*key[j] + key[j+1];

    sum = sum % 11 ;
    return sum;
}
```

dictionary.txt	
array	dizi
binary	ikili
child	cocuk
circuit	devre
class	sinif
client	istemci
gate	kapi
root	kok

2. Yukarıda dictionary.txt'de verilen kelimeleri Hash() fonksiyonunu ve çakışma çözümü olarak linear probing'i kullanarak relative.txt'ye yazınız. Ayrıca synonym chaining yöntemine göre bağlı listelere ilgili kayıtları ekleyiniz. (30P)

relative.txt		a-97	n-110
0	root kok	b-98	o-111
1	binary ikili	c-99	p-112
2		d-100	q-113
3		e-101	r-114
4		f-102	s-115
5		g-103	t-116
6		h-104	u-117
7	circuit devre	i-105	v-118
8	array dizi	j-106	w-119
9		k-107	x-120
10	child cocuk	l-108	y-121
ASCII Tablo →		m-109	z-122



```

bool empty()
{
    return (header->next == trailer);
}

```

```

void addFront(const int& i)
{
    add(header->next, i);
}

```

```

void add(DoublyNode* v, int& i)
{
    DoublyNode* u = new DoublyNode;
    u->score = i;
    .....
    .....
    .....
    .....
}

```

```

void main()
{
    DoublyLinkedList list;
    list.addFront(750);
    list.addFront(720);
}

```

3. Yukarıdaki programda add() fonksiyonunda ile temsil edilen satırlar:

i) (20P)

(Yanlış cevaptan 5P kırılacaktır)

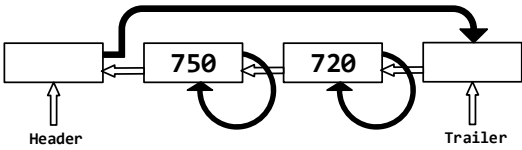
```

v->prev = u;
u->prev = v->prev;
v->prev->next = u;
u->next = v;

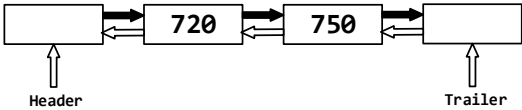
```

olduğunda listenin son hali :

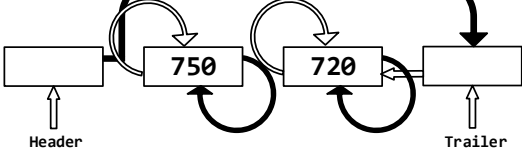
(A)



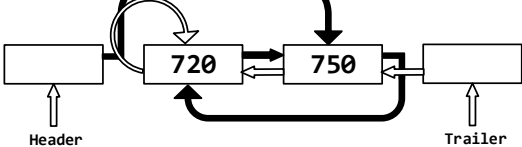
(B)



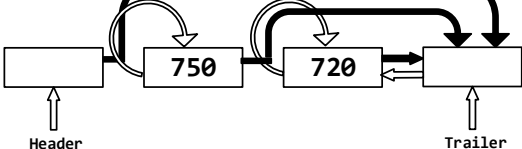
(C)



(D)



(E)



ii) (20P)

(Yanlış cevaptan 5P kırılacaktır)

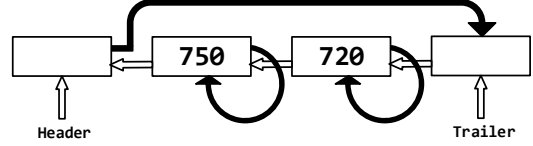
```

v->prev->next = u;
v->prev = u;
u->next = v;
u->prev = v->prev;

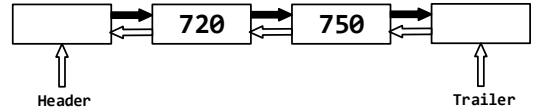
```

olduğunda listenin son hali :

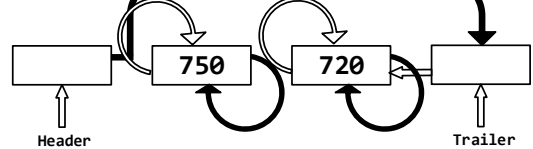
(A)



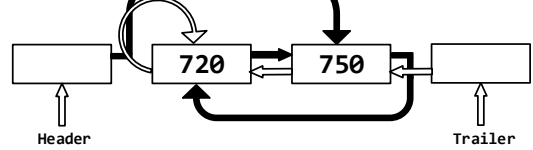
(B)



(C)



(D)



(E)

