**Öğrencinin**

| | |
|---|---|
| Adı Soyadı | |
| Numarası | |
| Dersi Aldığı Grup | |
| Öğrenim Türü | |

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
2012-2013 GÜZ DÖNEMİ VERİ YAPILARI DERSİ VİZE SINAVI

**SORU 1: Sabit boyutlu** bir dairesel kuyruğun şablon sınıfının prototipi aşağıda verilmiştir. (Fonksiyonlara ait gövdelerin yazıldığı düşünülecektir). Buna göre **main** fonksiyonu içerisindeki kod derlendiğinde **ekran çıktısı ne olacaktır**. (gerekli kütüphanelerin eklendiği düşünülecektir) **(10 Puan)**

**

```
const int MAX = 10;
template<typename T>
class Queue{
public:
   Queue();
   void  add(T item);       //kuyruğa eleman ekler
   bool  remove(T& item);  //kuyruktan eleman çıkartır
   int   count();         //kuyruktaki eleman sayısını getirir
private:
   T     m_Datas[MAX];
   int   m_iHead;
   int   m_iTail;
   int   m_iCount;
};
```

ALFABE: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

```
int main(int argc,char** argv){

 char temp;
 char yazi[]="DENEMELERRRRR";
 Queue<char>  kuyruk1,kuyruk2;

 for(int i=0;yazi[i]!='\0';i++)
       kuyruk1.add(yazi[i]);
 for(int i=0;i<5;i++){
       kuyruk1.remove(temp);
       kuyruk2.add(temp+i);
 }
 while(kuyruk2.remove(temp))
       kuyruk1.add(temp);
 while(kuyruk1.remove(temp))
       cout<<temp;
}
```

ÇIKTI: ELERRDFPHQ

**SORU 2: Sabit boyutlu** bir yığına ait şablon sınıfının prototipi aşağıda verilmiştir. Bu yığına ait olan iki fonksiyonun gövdesi de prototipin sağında verilmiştir. Buna göre boş bırakılan(noktalı) kısımları fonksiyonlar doğru çalışacak şekilde tamamlayın. **(10 Puan)** *

```
template<typename T>
class Stack{
public:
  Stack();
  ~Stack();
  bool push(T item);
  bool pop(T& item);
  bool isEmpty();
  int  count();
private:
  T*   m_pDatas; //veri dizisi
  int  m_iTop;
  int  m_iSize; //eleman sayısı
};
```

```
template<typename T>
bool Stack<T>::push(T item){

  if(m_iTop >=m_iSize)

       return false;

  m_pDatas[m_iTop] = item;

  m_iTop++

  return true;
}
```

```
template<typename T>
bool Stack<T>::pop(T& item)
{
  if(m_iTop<=0)

       return false;

  item = m_pDatas[m_iTop-1];

  m_iTop--;

  return true;
}
```

**SORU 3:** Aşağıdaki Kod bloğunun algoritma karmaşıklığı nedir? **(10 Puan)** *

```
int str[n][n];
for(int k = 0; k < n; k++)
for(int i=0;i<n;i++)
       str[k][i]++;
```

CEVAP:

$O(n^2)$

- **Zorluk Derecesi:** Soruların sağında bulunan yıldızlar zorluk seviyesini göstermektedir. Seviyeler **tek yıldızdan** (en kolay) **beş yıldıza**(en zor) kadar değişmektedir. Soruları çözerken stratejinizi yıldızlara göre belirleyebilirsiniz.
- **CEVAPLAR SORU KÂĞIDINA YAPILACAKTIR. SORU KÂĞITLARI İADE EDİLECEKTİR.**
- **HER SORU KAĞIDINA İSİM, NUMARA, GRUP VE ÖĞRENİM BİLGİLERİ YAZILACAKTIR**

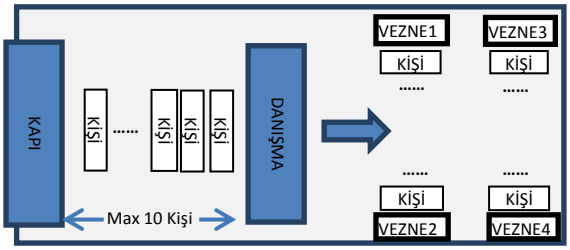
**SÜRE**

**90dk**

| Adı Soyadı Numarası | | Dersi Aldığı Grup Öğrenimi | |
|---|---|---|---|

**SORU 4:** Sekil 1'de bir bankanın müşterilerini veznelere yönlendirme mekanizması görünmektedir. **(15 Puan)**

- Bankaya giren müşteri öncelikle danışmaya gidip başvuru yapması gerekmektedir.(Danışma kuyruğuna eklenecek)
- Danışma, sıradaki müşteriyi dört vezneden kuyruğundaki müşteri sayısı en az olan vezneye yönlendirmektedir. (Danışma kişiyi uygun veznenin kuyruğuna ekleyecek ) ***VeznelereMusteriYolla()*** fonksiyonu.
  - Kuyruğunda en az elemanı bulunan vezne sayısı, birden fazla ise numarası en büyük olan veznenin kuyruğuna eklenecektir. (Vezne 3'ün numarası Vezne 2 den büyüktür).
  - DANIŞMA sınıfının kurucu fonksiyonu bütün kuyrukları oluşturmaktadır. Bu fonksiyonun yazılmış olduğu varsayılacaktır. Vezne kuyruklarına ait olan işaretçiler, DANISMA sınıfının prototipin de verilmiştir.
  - Kuyruk sınıfı olarak prototipi **Soru 1** de verilen Dairesel Kuyruk Sınıfı Kullanılacaktır.
  - Vezne kuyruklarının hepsi dolu ise danışma kuyruğundan kimse çıkartılmayacaktır. ****
- **Buna göre prototipi verilen DANISMA sınıfına ait fonksiyonların gövdelerini <u>bu kâğıdın arkasına</u> yazınız.**



Şekil 1

```
class DANISMA{
public:
   Queue<MUSTERI>*  pDanismaKuyruk;
   Queue<MUSTERI>*  pVezne1Kuyruk;
   Queue<MUSTERI>*  pVezne2Kuyruk;
   Queue<MUSTERI>*  pVezne3Kuyruk;
   Queue<MUSTERI>*  pVezne4Kuyruk;
   void VeznelereMusteriYolla();
   void DanismaKuyrugunaAl(MUSTERI m);
};
```

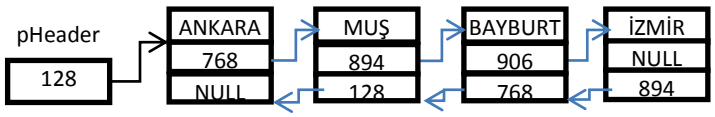
```
class MUSTERI
{
public:
  string ad;
  string soyAd;
}
```

**SORU 5:** Aşağıda iki yönlü bir bağlı listenin düğümleri ve bağlantıları verilmiştir. Listenin ilk düğümünün adresi **pHeader** işaretçisi içerisinde saklanmaktadır. Buna göre listeyi bir durumundan sonraki duruma getiren kodları sağ taraftaki boş satırlara yazıp, düğümlere ait olan noktalı kısımları doğru değerler ile doldurunuz. Düğümler **dinamik olarak** (heap alanında)oluşturulmuştur bu yüzden çöp oluşmamalıdır. Düğüm sınıfı aşağıda verilmiştir. **(20 Puan)**

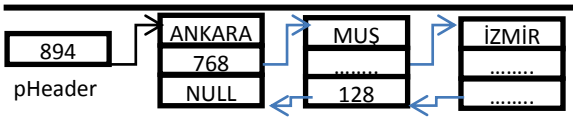
- Her bir satıra tek bir komut yazılmalı ve satır sayısı aşılmamalıdır.

***

```
template<typename T>
class DNode{
public:
        DNode (T a):Data(a),pNext(NULL),pPrev(NULL){}
        T       Data;
        DNode <T>*      pNext,pPrev;
};
```

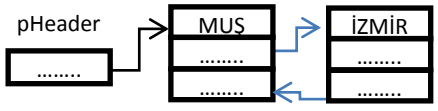



| | |
|---|---|
| 1.Satır | `DNode<string>* pTemp = pHeader;` |
| 2.Satır | `while(pTemp->pNext->pNext!=NULL)` |
| 3.Satır | `pTemp = pTemp->pNext;` |
| 4.Satır | `pTemp->pPrev->pNext = pTemp->pNext;` |
| 5.Satır | `pTemp->pNext->pPrev = pTemp->pPrev;` |
| 6.Satır | `delete pTemp;` |



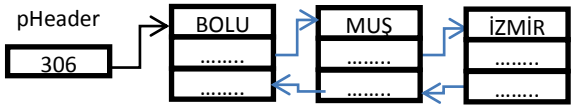

2.DURUM

| | |
|---|---|
| 7.Satır | `pHeader = pHeader->pNext;` |
| 8.Satır | `delete pHeader->pPrev;` |
| 9.Satır | `pHeader->pPrev = NULL;` |



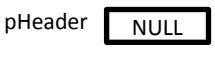

3.DURUM

| | |
|---|---|
| 10.Satır | `pHeader->pPrev = new DNode<string>("BOLU");` |
| 11.Satır | `pHeader = pHeader->pPrev;` |




4.DURUM

| | |
|---|---|
| 12.Satır | `while(pHeader->pNext!=NULL)` |
| 13.Satır | `{` |
| 14.Satır | `pHeader = pHeader->pNext;` |
| 15.Satır | `delete pHeader;` |
| 16.Satır | `}` |
| 17.Satır | `delete pHeader;` |
| 18.Satır | `pHeader = NULL;` |




5.DURUM

| Adı Soyadı | | Dersi Aldığı Grup | |
|---|---|---|---|
| Numarası | | Öğrenimi | |

**SORU 6:** Aşağıdaki fonksiyon, adresi kendisine verilen yığından istenilen bir elemanı çıkarmaya yaramaktadır. **(15 Puan)**

- Fonksiyonun ilk parametresi elemanın çıkartılacağı yığının adresini alacaktır.
- Fonksiyonun ikinci parametresi yığından çıkartılacak olan elemandır.
- Eğer aranan elemandan yığında birden fazla varsa hepsi çıkartılmalıdır
- Her kod satırı tablo içerisine yazılmalıdır.
- Stack(Yığın) sınıfının prototipi aşağıda verilmiştir. Sınıfa ait fonksiyonların gövdesi yazılmış olduğu varsayılacak.

**

```
template<typename T>
void  removeItemFromStack(Stack<T>* pStack,T findItem)
{
```

| | |
|---|---|
| 1. | `Stack<T>  tempStack;` |
| 2. | `T  tempItem;` |
| 3. | `while(pStack->pop(tempItem))` |
| 4. | `{` |
| 5. | `  if(tempItem ==findItem)` |
| 6. | `  {` |
| 7. | `  continue;` |
| 8. | `  }` |
| 9. | `  tempStack.push(tempItem);` |
| 10. | `}` |
| 11. | `while(tempStack.pop(tempItem))` |
| 12. | `  pStack->push(tempItem);` |

```
}
```

```
template<typename T>
class Stack
{
public:
        Stack();
        ~Stack();
        bool    push(T item);
        bool    pop(T& item);
        bool    isEmpty();
        int     count();
private:
        T*      m_pDatas;
        int     m_iTop;
        int     m_iSize;
};
```

**SORU 7:** **Aşağıdaki** program çalıştırıldığında ekrana ne yazacağını ***cout*** *metodunun* hemen karşısına yazınız. (Kod bir bütün olarak değerlendirilmelidir.) **(10 Puan)**

**

```
int main()
{
        int x=100;
        int *xptr=&x;
        Node<int> *p = new Node<int>(x);
        (*xptr)++;
        Node<int> *r = new Node<int>(x);
        Node<int> **ptr=&r;
        if(ptr==&r)(*xptr)++;
        if(*ptr==r)(*xptr)++;
```

| | |
|---|---|
| `cout<<x<<endl;` | 103 |
| `cout<<*xptr<<endl;` | 103 |
| `cout<<p->eleman<<endl;` | 100 |
| `cout<<r->eleman<<endl;` | 101 |
| `cout<<(*ptr)->eleman<<endl;` | 101 |

```
        delete r;
        delete p;
}
```

**SORU 8:** Aşağıdaki infix ifadeyi stack kullanarak postfix'e çeviriniz. Cevabı bu kâğıdın arkasına yazınız.(Şekil üzerinde gösteriniz.) **(10 Puan)**

(5+3)*(8-5)/6-2

*

```
void   DANISMA::VeznelereMusteriYolla()

{

	//bütün kuyruklar dolu ise fonksiyon hiç bir iş yapmadan dönecektir.
	if(pVezneKuyruk2->count()==10
		&&pVezneKuyruk2->count()==10
		&&pVezneKuyruk2->count()==10
		&&pVezneKuyruk2->count()==10)
	{
		return;
	}
	int temp;
	if(pDanismaKuyruk->remove(temp))
		return;

	Queue<int>*   pKuyruk = pVezneKuyruk1;

	int count = pVezneKuyruk1->count();

	if(count>=pVezneKuyruk2->count())
	{
		count = pVezneKuyruk2->count();
		pKuyruk = pVezneKuyruk2;
	}
	if(count>=pVezneKuyruk3->count())
	{
		count = pVezneKuyruk3->count();
		pKuyruk = pVezneKuyruk3;
	}
	if(count>=pVezneKuyruk4->count())
	{
		count = pVezneKuyruk4->count();
		pKuyruk = pVezneKuyruk4;
	}

	pKuyruk(temp);
}
```

| ( | 5 | + | 3 | ) | * | ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

PostFix =

| 5 | + | 3 | ) | * | ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

STACK

| |
|---|
| ( |

PostFix =

| + | 3 | ) | * | ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

STACK

| |
|---|
| ( |

PostFix = 5

| 3 | ) | * | ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|

STACK

| + |
|---|
| ( |

PostFix = 5

| ) | * | ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|---|---|

STACK

| + |
|---|
| ( |

PostFix = 53

| * | ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|---|

STACK

| |
|---|
| |

PostFix = 53+

| ( | 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|---|

STACK

| |
|---|
| * |

PostFix = 53+

| 8 | - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|---|

STACK

| ( |
|---|
| * |

PostFix = 53+

| - | 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|---|

STACK

| ( |
|---|
| * |

PostFix = 53+8

| 5 | ) | / | 6 | - | 2 |
|---|---|---|---|---|---|

STACK

| - |
|---|
| ( |
| * |

PostFix = 53+8

| ) | / | 6 | - | 2 |
|---|---|---|---|---|

STACK

| - |
|---|
| ( |
| * |

PostFix = 53+85

| / | 6 | - | 2 |
|---|---|---|---|

STACK

| * |
|---|

PostFix = 53+85-

| 6 | - | 2 |
|---|---|---|

STACK

| / |
|---|

PostFix = 53+85-*

| - | 2 |
|---|---|

STACK

| / |
|---|

PostFix = 53+85-*6/

| 2 |
|---|

STACK

| - |
|---|

PostFix = 53+85-*6/

PostFix = 53+85-*6/2-