İSTANBUL TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM FAKÜLTESİ
Date: 01.11.2011

Student Id :	....
Name Surname :	....
Signature :	....

DATA STRUCTURES

MIDTERM 1

Question	1	2	3	4	5	Result
Point						

1)[5 Points] What is the output of the following program fragment?

```
        char Text[9] = "Sandrine";
        for(char *Ptr = Text; *Ptr != '\0'; ++Ptr)
        cout << Ptr << endl;
```

Output of the Program:

```
Sandrine
andrine
ndrine
drine
rine
ine
ne
e
```

2)[5 Points] Suppose the following definitions are in effect.

```
        int a[10], b[10], int c;
        int* d;
```

Which of the following assignments are legal (just mark statements as “legal” or “illegal”)?

```
        b = a;        _______illegal_________
        c = a[1];     _______legal _________
        d = a;        _______legal_________
        d = &c;       _______legal_________
        *d = b;       _______illegal_________
        *d = b[9];    _______legal_________
```

3)[15 Points] Find the problematic line and fix it by commenting it out. Then, write down the output of the modified code.

```
#include<iostream>
using namespace std;

void main(){
        int a, b, c;
        b = c = 5;
        int* d, *e;
        int *f;
        //*d = b; //CANNOT WRITE PRIOR TO INITIALIZATION
        e = &c;
        cout << (e==&b) << endl;
        cout << (d==&b) << endl;
        cout << a << endl;
        cout << b << endl;
        cout << c << endl;
        cout << d << endl;
        f = &a;
        cout << *f << endl;
        *e = 7;
        d = &b;
        cout << c << endl;
        cout << &c << endl;
        cout << *d << endl;
        cout << b << endl;
        cout << e << endl;
        cout << &e << endl;
}
```

Output of the Program:

```
0
0
-858993460
5
5
CCCCCCCC
-858993460
7
0034FC40
5
5
0034FC40
0034FC28
```

4)[25 Points] A linked list is used to store letters. The given functions do the following:

- "add" function adds a single letter at the end of the linked list.
- "addAll" function adds an array of letters to the end of the linked list.
- "emptyList" function creates and returns an empty linked list. The empty list has a single node. This node points NULL and stores '\0'.
- "print" function prints out the letters of the linked list in order.
- "deleteList" deletes every element in the list.
- The main function creates a linked list of letters of the sentence "istanbul technical university". Then, splits the sentence to its words, capitalize the beginning of each word, print the word out and delete the word from memory.

Implement these two missing functions:

- "replace" function that replaces each occurrence of a given character(second parameter) to another(third parameter).
- "split" function that splits the linked list from the first occurrence of the given character. Assume that, the beginning or end of a linked list cannot contain the character used to split the linked list. (So, delete the node storing the split character while splitting.) Split function must return a pointer to the first node after the split character. If there is no split character stored in the linked list, it must return NULL.

According to the given explanation, the output of the program must look like below:

```
Istanbul
Technical
University
```

```
#include<iostream>
using namespace std;

struct node{
        char data;
        node* next;
};
```

```
list* emptyList(){
        node* l = new node;
        l->data = '\0';
        l->next=NULL;
        return l;
}
```

```
typedef node list;


void replace(list* l, char o, char n)
{
do{
l->data = l->data == o ? n : l->data;
l=l->next;
}while(l!=NULL);









}
list* split(list* l, char c){
while(l->next!=NULL){
if (l->next->data==c){
      node* r = l->next->next;
      delete l->next;
      l->next = NULL;
      return r;
      }
            l=l->next;
      }
      return NULL;




}

void print(list* l){
      node* cursor = l;
      while(cursor->next!=NULL){
            cout << cursor->data;
            cursor=cursor->next;
      }
      cout << cursor->data << endl;
}

void addAll(list* l, char* c){
      while(*c!='\0') add(l, *(c++)); }
```

```
void add(list* l, char c){

if(l->next==NULL && l->data=='\0'){
            l->data = c;
            return;
      }
while(l->next!=NULL) l=l->next;
      node* nn = new node;
      nn->next = NULL;
      nn->data = c;
      l->next = nn;
}

void deleteList(list* l){
  node* n = l;
  while(l->next!=NULL){
     while(n->next->next!=NULL)
           n=n->next;
            delete n->next;
            n->next = NULL;
            n=l;
      }
      delete l;
}



void main(){
      list* l = emptyList();
      addAll(l, "istanbul teknik 
universitesi");
      list* k = l;
      do{
      l=split(k, ' ');
replace(k, k->data, k->data-32);
      print(k);
      deleteList(k);
      k = l;
      }while(k!=NULL);
}
```

5)[50 Points] Information of a group of course: course name, course day and course time will be stored in a linked list structure. Course day may be a day between “Monday”-“Friday” and course time may be a discrete variable which has a meaning either “BeforeMidday” or “AfterMidday”(Numeric time will not be used). There should be more than one course at the same day and the same time.

a) Write the necessary definitions to construct the linked list structure.

b) Assume courses are ordered randomly in a list. You are expected to write a function which takes the unsorted list as a parameter and print the courses at the specified day and the specified time. The function should take three parameters: pointer of the list, day, time. There should not be a return parameter.

c) Write a function which takes the unsorted list as a parameter and creates a new list. The unsorted list should not be changed. The new list should be sorted according to the course days and course times in the day.

The sorting procedure is given below:

1) Read the information of a course from the list.
2) Add that course as a new node in an appropriate place(head/middle/last) to the second list. Function should return a pointer to the second(sorted) list.
3) Repeat the procedure for each node.

a)
```
struct ders{
    char ad[20];
    int gun;//1-7
    int zaman;//1-2
    ders *sonraki;
};
struct liste{
    ders *bas;
};
```

b)
```
void listele(liste *dl,int gun,int zaman){
    ders *tara=dl->bas;
    while(tara!=NULL){
        if(tara->gun==gun && tara->zaman==zaman){
            cout<<tara->ad<<endl;
        }
        tara=tara->sonraki;
    }
}
```

c)
```
    liste * siralilisteolustur(liste *derslistesi){
    ders *tara=derslistesi->bas;
    liste *sıraliliste=new liste;
    ders *yeni;
    siraliliste->bas=NULL;

    while(tara){//orijinal ders listesi üzerinde döner.
        yeni=new ders;
        strcpy(yeni->ad,tara->ad);
        yeni->gun=tara->gun;
        yeni->zaman=tara->zaman;
        //basa ekleme
        if(siraliliste->bas==NULL ||
               tara->gun<siraliliste->bas->gun ||
              (tara->gun==siraliliste->bas->gun && tara->zaman<=siraliliste->bas->zaman)){
              yeni->sonraki=siraliliste->bas;
              siraliliste->bas=yeni;
        }


        else{
            ders *t=siraliliste->bas;//doğru yeri bulmak için yeni listede arama
            ders *arka=t;
            while(t && tara->gun >t->gun){
                arka=t;
                t=t->sonraki;
            }




          while(t && tara->gun==t->gun && tara->zaman>t->zaman){
              arka=t;
              t=t->sonraki;
            }
            arka->sonraki=yeni;
            yeni->sonraki=t;
        }
        tara=tara->sonraki;
    }
    return siraliliste;
}
```