ANKARA UNIVERSITY
COMPUTER ENGINEERING
COM2067/COM267 MIDTERM
180 m.
(Write answers of each question on same pdf file. Please submit only one file in which all 3 answers are placed.)

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

SINAVLAR VE ÖDEVLER İÇİN ŞEREF SÖZÜ

Bir Ankara Üniversitesi öğrencisi olarak;

- -Bu ödevde/sınavda yardım almadığımı ya da hiç kimseye yardım etmediğimi,
- -Başkasına ait olan bir çalışmayı kendi çalışmam olarak sunmadığımı,
- -Sınav/ödev sorularının çözümü için hiç kimseden (öğrenci, öğretim üyesi ya da arkadaş) yardım istemediğimi,
- -Problemin çözümünü bulmak için interneti ya da çevrimiçi ya da basılı herhangi bir belgeyi kullanmadığımı beyan ederim.

Yukarıdaki ifadelere uymadığımın tespit edilmesi durumunda sınavdan/ödevden sıfır alacağımı ve hakkımda **Ankara Üniversitesi Öğrenci Disiplin Yönetmeliği** çerçevesinde soruşturma açılacağını biliyorum.

HONOR CODE FOR EXAMS and ASSIGMENTS

As an Ankara University student, I agree that;

- I have neither given nor received unauthorized assistance on this exam or assignment.
- I have not represented the work of another as my work.
- I have not asked someone else (student, teacher, and friends) to help with this assignment or exam questions.
- I have not used the internet or any online or printed document to find problem solutions

I understand that failure to comply with the statements above will result in receiving a zero from this exam/assignment and being reported for academic dishonesty by the **disciplinary policies of Ankara University**.

1. (30 p) Explain briefly what the following functions that use the node structure below do.

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

(a)

```
void function1(struct node** a, struct node** b) {
   struct node* cur;
   if (*a == NULL) {
      *a = *b;
   }
   else {
      cur = *a;
      while (cur ->next != NULL) {
         cur = cur ->next;
      }
      cur ->next = *b;
   }
   *b=NULL;
}
```

(b)

```
static void function2(struct node** head) {
   struct node* r = NULL;
   struct node* cur = *head;
   struct node* n;
   while (cur!= NULL) {
      n = cur ->next;
      cur ->next = r;
      r = cur;
      cur = n;
   }
   *head = r;
}
```

(c)

```
void function3(struct node* head) {
   struct node* cur = head;
   if (cur == NULL) return;
   while(cur ->next!=NULL) {
      if (cur ->data == cur ->next->data) {
         struct node* nextNext = cur ->next->next;
         free(cur ->next);
         cur ->next = nextNext;
      }
      else {
         cur = cur ->next;
      }
   }
}
```

2. (35 p) Write two different C functions one takes a pointer that represents the head of a linked list (a node has an integer value and nextptr), the other takes a structure array (each structure has an integer value as a member). The second function has also a parameter that shows the size of the array. (Both the linked list and the array are not sorted.) (Those functions are independent of each other.)

```
struct structForLinkedList{
        int value;
        struct structForLinkedList* nextptr;
};
struct structForArray{
        int value;
};
void computeDisplayFrequenciesLinkedList(struct structForLinkedList* head);
void computeDisplayFrequenciesArray(struct structForArray myArray[], int size);
```

Each function computes and displays how many times each integer value repeats in the linked list and in the structure array respectively. The output should be presented in decreasing order with respect to frequencies of the integers. (If the frequencies are same, the greater integer placed first). Do not write a complete C program. Write only two function definitions.

Example Output: (There will be seven 27s in the list (linked list or array), six 38s, five 45s and ...).

27 7
38 6
45 5
40 5
23 3
55 2
21 2
11 1
9 1

3. (35 p) Define a node structure for the structure given below.

```
struct node{

   ...

};
```

Write a complete C program that you will create the connections between nodes. You can assign, 1 to the value of the 1st element, 2 to the value of the 2nd element, n to the value if the nth element. Assume that the value n is user-entered in your program.