

1) Write the missing "add" and "empty" functions for the given circular doubly linked list. The "add" function should always insert to the beginning of the list. Please make sure that the list should convey the requirements for a circular doubly linked list after each change.

```
struct Node{
    int data;
    Node *next;
    Node *previous;
};

struct CDLL{
    Node *final;
    void create();
    void empty();
    void add(Node *newptr);
};

void CDLL::create(){
    final=NULL;
}
```

```
int main(){
    Node newnode={9,NULL,NULL};
    CDLL cdll;
    cdll.create();
    cdll.add(&newnode);
    for(int i=8;i>=0;i--){
        newnode.data=i;
        cdll.add(&newnode);
    }
    cdll.empty();
}
```

```
void CDLL::add(Node *newptr){  
    Node *newNode=new Node;  
    newNode->data=newptr->data;  
    if(final==NULL){  
        newNode->next=newNode;  
        newNode->previous=newNode;  
        final=newNode;  
    } else {  
        Node *oldHead=final->next;  
        newNode->next=oldHead;  
        oldHead->previous=newNode;  
        newNode->previous=final;  
        final->next=newNode;  
    }  
};
```

```
void CDLL::empty(){
    final->previous->next=NULL;
    Node * n;
    while(final){
        n=final;
        final=final->next;
        delete n;
    }
};
```

2) The below single link list's nodes consist of a name string and a dynamic integer array for holding the grades.

- In its current state something is wrong with the “add” function.
- The state of the link list after each insert operation (`sl.add(&newnode);`) is given below and shows that the insert operation is not working properly.
- Please explain the reason why this situation occurs.
- And propose the correct solution together with required changes/additions to the “add” method.
- (“add” function should still add to the beginning of the linked list)

```

struct Student{
    char *name;
    int *grades;
    Student *next;
};
struct StudentList{
    Student *head;
    void create();
    void add(Student *newptr);
};
void StudentList::create(){
    head=NULL;
}
void StudentList::add(Student
                      *newptr) {
    Student *n=new Student();
    *n=*newptr;
    if(head==NULL) {
        head=n;
        return;
    }
    n->next=head;
    head=n;
}

```

```

int main(){
    StudentList sl;
    sl.create();
    char *name="Ali"; int grades[]={20,50,80};
    Student newnode=(name,grades,NULL);
/*A1*/sl.add(&newnode);

    name="Veli"; grades[0]=100; grades[1]=100;
    grades[2]=100;
    newnode.grades=grades; newnode.name=name;
/*A2*/sl.add(&newnode);

    char name2[]="mehmet"; newnode.name=name2;
/*A3*/sl.add(&newnode);

    strcpy(name2,"ahmet"); newnode.name=name2;
/*A4*/sl.add(&newnode);
}

```

After /*A1*/

sl	{head=0x003a1d68 }
head	0x003a1d68 {name=0x012b58a8 "Ali" ;}
+ name	0x012b58a8 "Ali"
+ grades	0x0024fe1c
	20

After /*A2*/

sl	{head=0x003a1db0 }
head	0x003a1db0 {name=0x012b58a0 "Veli" grades=0;}
+ name	0x012b58a0 "Veli"
+ grades	0x0024fe1c
	100
+ next	0x003a1d68 {name=0x012b58a8 "Ali" grades=0;}
+ name	0x012b58a8 "Ali"
+ grades	0x0024fe1c
	100
+ next	0x00000000 {name=??? grades=??? next=??? }

After /*A3*/

sl	{head=0x003e1df8 }
head	0x003e1df8 {name=0x001af738 "mehmet" ;}
+ name	0x001af738 "mehmet"
+ grades	0x001af75c
+ next	0x003e1db0 {name=0x002f58a0 "Veli" grades=0;}

After /*A4*/

sl	{head=0x003e1e40 }
head	0x003e1e40 {name=0x001af738 "ahmet" ;}
+ name	0x001af738 "ahmet"
+ grades	0x001af75c
+ next	0x003e1df8 {name=0x001af738 "ahmet" ;}

Fixing add method

```
void StudentList::add(Student *newptr){  
    Student *n=new Student();  
    n->name=new char[strlen(newptr->name)];  
    strcpy(n->name,newptr->name);  
    n->grades=new int[GRADE_COUNT];  
    for(int i=0;i<GRADE_COUNT;i++)  
        n->grades[i]=newptr->grades[i];  
    if(head==NULL){  
        head=n;  
        return;  
    }  
    n->next=head;  
    head=n;  
}
```

3) Aşağıda verilen kod için aşağıdaki şıkları cevaplayınız. Cevaplar sırasında ana fonksiyonun 3. Satırına gelindiğinde değişkenlerin bellekte yerleştirilecekleri adreslerin yukarıda verilen şekilde olduğunu düşününüz.

```
#include <iostream>
using namespace std;
void kar_bul(char *dizi,char **isr,char ch) {
    *isr=dizi;
    char *ptr=*isr;
    while(ptr && *ptr!=ch)
        ptr++;
    *isr=ptr;
}

int main() {
    char katar[10]="ahmet";
    char *ptr;
    kar_bul(katar,&ptr,'m');
    cout<<ptr<<endl;
    getchar();
    return EXIT_SUCCESS;
}
```

Değişkenlerin bellekteki yerleşimleri

[+]	8ptr	0x0024fd7c
[+]	katar	0x0024fd88 "ahme

c. Kar_bul fonksiyonunun son satırı yürütüldüğünde “*isr” değişkenin taşıdığı değer ne olacaktır?

ox0024fd8a

d. Bu program çalıştığında ekrana yazdırılacak çıktı nedir?

met

a. Kar_bul fonksiyonunun ikinci satırında "isr" değişkenin değeri nedir?

ox0024fd7c

e. Bu program bu örnek üzerinde doğru şekilde çalışmasına rağmen sonlanma koşulunda ufak bir problem vardır. Programı içerisinde 'm' harfi bulunmayan bir katar ile yeniden çalıştırarak hatanın nerede olduğunu belirtiniz.

```
#include <iostream>
using namespace std;
void kar_bul(char *dizi, char
```

While döngüsünde *ptr != '\0' olmalıdır katarın sonuna gelindiği bu şekilde kontrol edilir.

f. e şıkkındaki gerekli düzeltmeyi yaptıktan sonra içerisinde 'm' harfi içermeyen bir katar için program sonunda ekrana yazılacak çıktı ne olacaktır?

```
#include <iostream>
```

4) a. Aşağıda verilen kod parçası çalıştırılmak istendiğinde tek bir satırda hata vermektedir. Hatayı ve nedenini açıklayınız. Hatayı düzeltiniz.

b. Kod parçasının çıktısını yazınız.

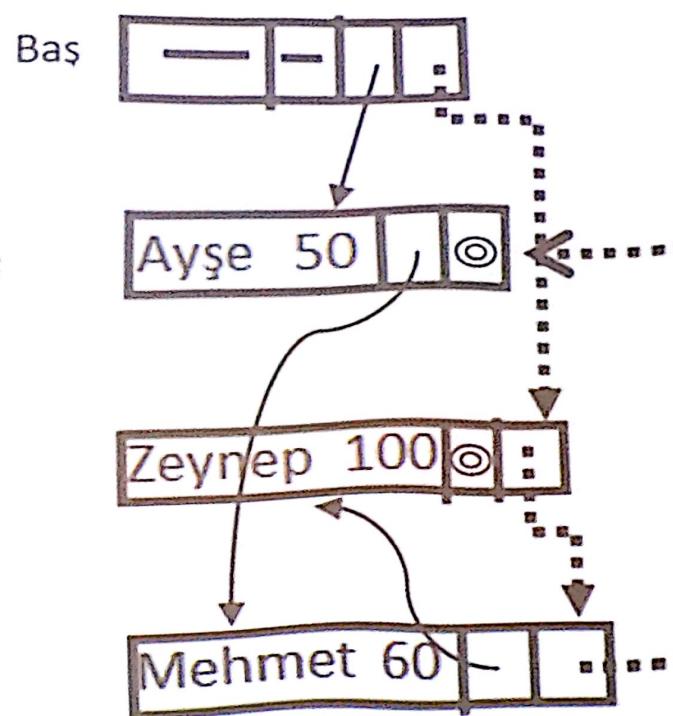
```
char katar[10];
katar="ahmet"; // HATA
char *ptr;
ptr=katar;
char *katar2="zeynep";
katar2=ptr;
strcpy(katar2,"ali");
cout<<katar<<endl;
```

2. satırda strcpy(katar,"ahmet"); olmalı
Dizi adı sabit işaretçidir. Atama yapılamaz.

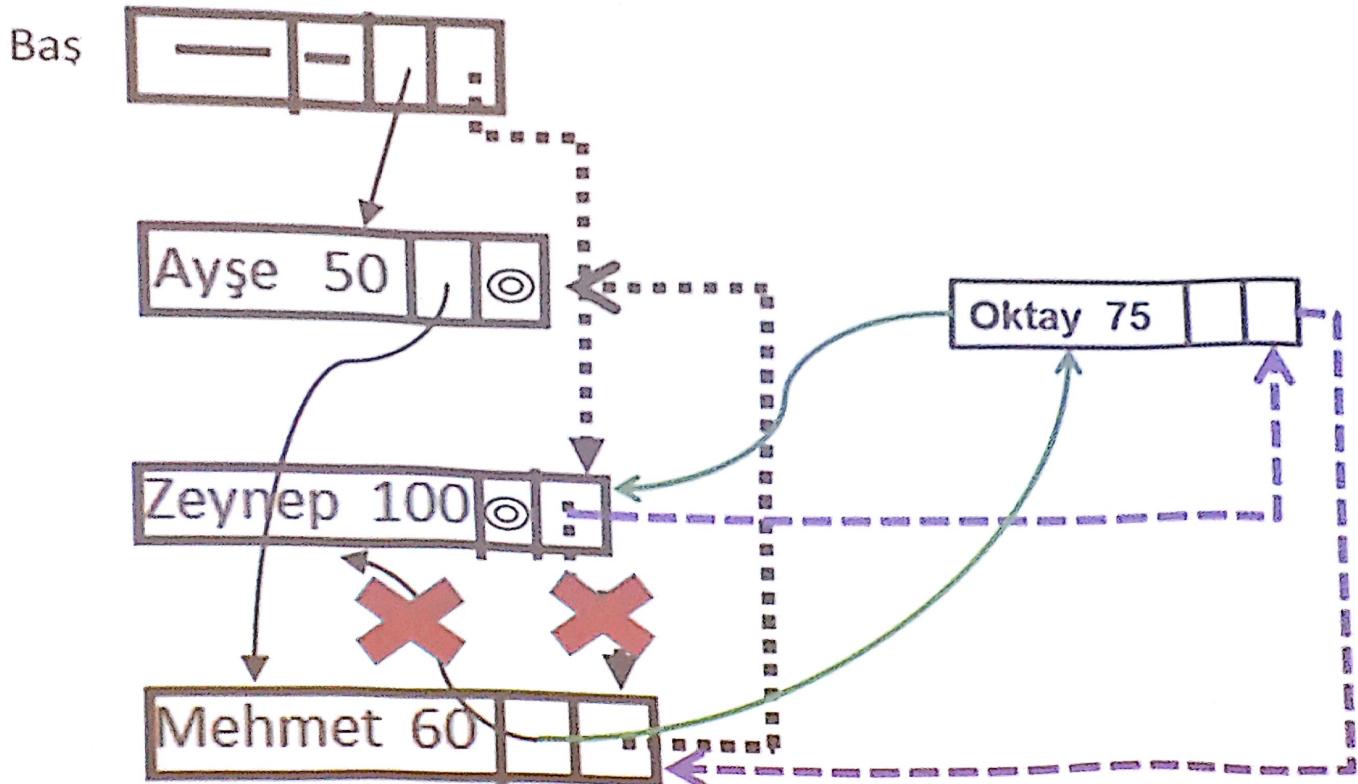
Cıktı: "ali"

5) Öğrencilerin bir sınavdan aldığı notlar bir liste yapısı üzerinde tutulmak istenmektedir. Bu yapının iki farklı kritere göre sıralı tutulması gerekmektedir.

- Bunlardan ilki ada göre artan sıradada, ikincisi ise nota göre azalan sıradadır.
- Listenin başlangıç düğümü iki farklı başlama noktasına referans içermek üzere boş bir düğümden oluşmaktadır.
- Bağlantılarda kullanılan düz çizgi ada göre sıralamayı, noktalı çizgi ise nota göre sıralamayı göstermektedir.



a) Listeye (Oktay,75) şeklinde bir kaydın eklenmesi durumunda oluşan yeni listeyi çiziniz.



b) Listenin düğüm yapısını yazınız.

```
struct Node{  
    char name[NAME_LENGTH];  
    int grade;  
    Node *next_name;  
    Node *next_grade;  
};
```

c) Listeye ekleme metodunu yazınız.

```
void add(char *name, int grade){  
    Node *traverse, *tail;  
    Node * node=new Node;  
    node->grade=grade;  
    strcpy(node->name,name);  
    node->next_name=NULL;  
    node->next_grade=NULL;  
    if(head->next_name==NULL){  
        head->next_name=node;  
        head->next_grade=node;  
        return;  
    }  
}
```

```
if (strcmp(node->name, head->next_name->name) < 0) {  
    node->next_name = head->next_name;  
    head->next_name = node;  
} else {  
    traverse=head->next_name;  
    while (traverse && (strcmp(node->name,traverse->name) > 0)){  
        tail = traverse;  
        traverse = traverse->next_name;  
    }  
    node->next_name = traverse;  
    tail->next_name= node;  
}
```

```
if ((node->grade) > (head->next_grade->grade)) {  
    node->next_grade= head->next_grade;  
    head->next_grade= node;  
} else {  
    traverse=head->next_grade;  
    while (traverse && (node->grade ) < (traverse->grade)){  
        tail = traverse;  
        traverse = traverse->next_grade;  
    }  
    node->next_grade= traverse;  
    tail->next_grade= node;  
}
```

d) İsim sırasında listeleyen fonksiyonu yazınız.

```
void list_by_name(){
    Node * traverse= head->next_name;
    while(traverse){
        cout<< traverse->name << endl;
        traverse=traverse->next_name;
    }
}
```