Soru 1

Aynı değerli içerebilecek bir dizi verildiğinde, tüm elemanları ve bunların frekanslarını yazın

Cevabı

122670604

mehmet fatih
Atas

```
#include <stdio.h>
int main()
{
int arr[max_size];
int freq[max_size];
int size, i, j, count;
Printf("Dizinin boyutunu girin:");
scanf("%d", &size);

Printf("Dizi elemanlarını girin:\n");
for (i=0, i<size; i++)
{
scanf("%d, &arr[i];
freq[i]=-1;
}
for(i=0; i<size; i++)
}
count=1;
for (j=i+1; j<size ; j++)
{
if(arr[i] == arr[j])
{
count ++;
freq[j]=0;
}
};
freq[i] !=0)
{
freq[i]=count;
}
}
Printf("\n Tüm elemanlar ve frekanslar:\n");
for(i=0; i<size; i++)
{
if(freq[i] !=0)
{
printf("%d occurs%d times\n", arr[i], freq[i];
}
}return 0;
}
```

Soru 2

İki sıralanmış dizi verildiğinde, onları sıralanmış bir şekilde birleştiren C kodunu yazın

Cevabı

```
#include <stdio.h>

void birlestir (int dizi1[], int len1, int dizi2[], int len2, int birlestirilmis[])
{
int i=0, j=0, k=0;
while (i<len1 && j<len2)
{
if(dizi1[i] <dizi2[j]) {
birlestirilmis [k++]=dizi1[i++];
} else {
birlestirilmis [k++] =dizi2 [j++];
}
}
while (i<len1) {
  birlestirilmis [k++]= dizi1 [i++];
}
while (j <len2) {
birlestirilmis [k++]=dizi2 [j++];
}
}
int main() {
int dizi1[] = {1,3,5,7,9};
int len1 = sizeof(dizi1) / sizeof(int);
int dizi2[] = {2,4,6,8,10};
int len2=sizeof(dizi2) \ sizeof(int);
int birlestirilmis [len1 + len2];
birlestir(dizi1, len1, dizi2, len2, birlestirilmis);
for(int i=0; i<len1 +len2; i++) {
printf("%d, birlestirilmis [i]);
}
Printf("\n");
  return 0;
}
```

mehmet
fatih
Ataş

122670604

Soru 3

İki sıralanmış bağlantılı listeyi birleştirin ve yeni bir sıralanmış listeye döndürün, yeni listede iki listenin birleşik hali olmalı.

Cevap

mehmet
fatih
Ataş
1226706004

```
#include <stdio.h>
#include <stdlib.h>

typedef struct listNode{
int val;
 struct listNode *next;
} ListNode;

listNode *birlestir(listNode*l1,listNode*l2){
 listNode birlestirilmis={0}, *current = &birlestirilmis;

while(l1 && l2){
if(l1->val<l2->val){
   current->next=l1;
l1=l1->next;
}else{
 current->next=l2;
 l2=l2->next;
}
 current=current->next;
}
current->next =l1 ? l1:l2;
return birlestirilmis.next;
}
 int main(){
listNode l1[]={{1,&l1[1]},{2,&l1[2]},{4,Null}};
listNode l2[]={{1,&l2[1]},{3,&l2[2]},{4,Null}};

listNode *birlestirilmis = birlestir(l1,l2);

while(birlestirilmis){
printf("%d ", birlestirilmis->val);
 birlestirilmis =birlestirilmis->next;
}
return 0;
}
```

Soru 4'ün cevabı

mehmet fatih Atas
122670 6004

```
#include <stdio.h>
#include<stdlib.h>
// bağlantılı listelerin düğüm yapısı struct Node{
int val;
struct Node *next;
};
// yeni düğüm oluşturma fonksiyonu
struct Node* newNode(int val){
struct node *temp =(struct Node*)malloc(sizeof(struct Node));
temp->val=val;
temp->next=Null;
return temp;
}
// iki bağlantılı listein toplamını hesaplayan fonksiyon
 struct Node *addlists(struct Node *l1, struct Node *l2) {
 struct node *res=Null;
 struct Node *temp, *prev=null;
 int carry =0, sum;
while (l1!=Null||l2!=Null){
Sum=carry +(l1?l1->val:0)+(l2?l2->val:0);
carry=(sum>=10)?1:0;
 Sum=sum%10;
 temp=New node(sum);
 if(res==null){
  res=temp;
 }
else{
  prev->next=temp;
 }
 prev=temp;
 if(l1){
   l1=l1->next;
  }
  if(l2){
   l2=l2->next
  }
 }
```

```
if(carry>0){
   temp->next=newNode(carry);
}
 return res;
}
```

Soru 5'in cevabı

mehmet fatih Ataş
1226706004

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 100
typedef struct{
char data[MAX_SIZE];
 int front,rear;
}Queue
Queue createQueue(){
Queue q=(Queue)malloc(sizeof(Queue));
 q->front=q->rear=-1;
 return q;
}
int isfull(Queue q){
return (q->rear==MAX_SIZE-1);
}
int isEmpty(Queue q){
 return (q->front==-1 q->front q->rear);
}
void enqueue(Queue q, char item){
 if(isfull(q)){
 Printf(Queue is full.n);
 return;
 }
q->data[++q->rear]=item;
 if(q->front==-1)q->front=0;
}
char dequeue(Queue q){
 if(isempty(q)){
 Printf(Queue is empty.n);
 return'0';
 }
char item=q->data[q->front++];
 if(q->front q->rear)q->front=q->rear=-1;
 return item;
 }
```

```
void remove dublicates(Queue q){
if(isEmpty(q)return;
char prev='0';
while(!isEmpty(q)){
char current=dequeue(q);
if (current!=prev){
 Prev=current
 enqueue(q,current);
   }
  }
}
int main(){
Queue q=create Queue();
char str[MAX_SIZE];
Printf(Enter a string);
scanf(%s,str);
for(int=0; str[i]!='0';i++)
enqueue(q,str[i];
}
remove dublicates(q);
Printf(Result);
while(!isEmpty(q));
 Printf(%c,dequeue(q));
}
Printf(n);
return 0;
```