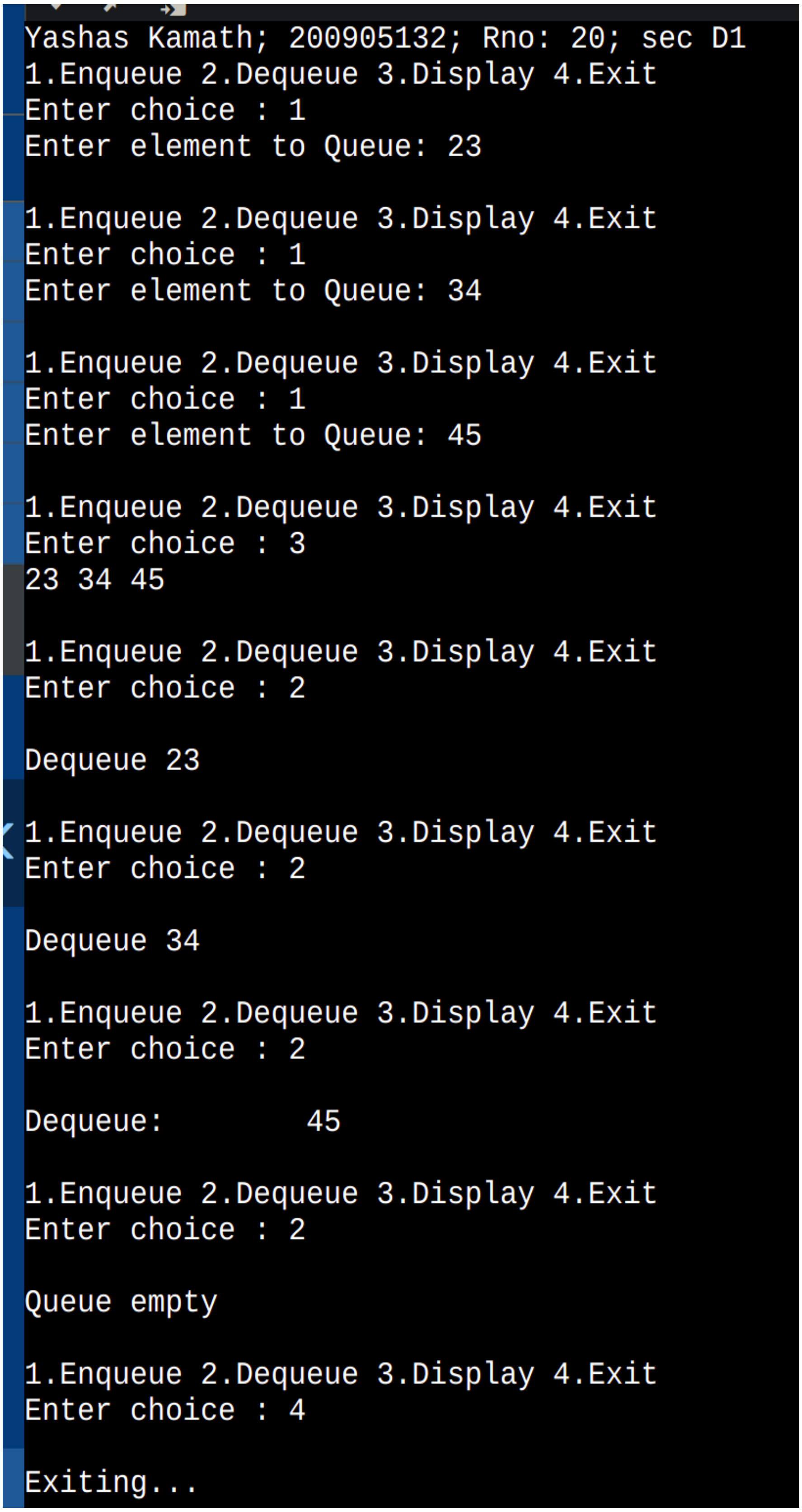
DSA LAB 7

Q1) Implement a queue using singly linked list without header node.

Solution:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
int data;
struct node * next;
} * NODE;
NODE enqueue(NODE first, int ele){
NODE temp = (NODE)malloc(sizeof(struct node));
temp->data = ele;
if(first == NULL){
return temp;
}
else{
NODE m = first;
while(m->next != NULL){
m=m->next;
}
m->next = temp;
return first;
}
}
NODE dequeue(NODE first){
if(first == NULL){
printf("\nQueue empty\n");
return NULL;
}
else if(first->next == NULL){
printf("\nDequeue:\t%d\n",first->data);
free(first);
return NULL;
}
else{
NODE temp = first;
first = first->next;
printf("\nDequeue\t%d\n",temp->data);
free(temp);
return first;
}
}
void display(NODE first){
if(first == NULL){
printf("\nQueue empty\n");}
else{
NODE temp = first;
while(temp->next != NULL){
printf("%d ",temp->data);
temp=temp->next;
}
printf("%d\n",temp->data);
}
}
int main(){
    printf("Yashas Kamath; 200905132; Rno: 20; sec D1");
NODE first = NULL;
int ch,ele;
while(1){
printf("\n1.Enqueue 2.Dequeue 3.Display 4.Exit\nEnter choice : ");
scanf("%d",&ch);
switch(ch){
case 1: printf("Enter element to Queue: ");
scanf("%d",&ele);
first = enqueue(first,ele);
break;
case 2: first = dequeue(first);
break;
case 3: display(first);
break;
case 4: printf("\nExiting...");
return 0;
}
}
}
```



```
Yashas Kamath; 200905132; Rno: 20; sec D1
1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 1
Enter element to Queue: 23

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 1
Enter element to Queue: 34

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 1
Enter element to Queue: 45

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 3
23 34 45

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 2

Dequeue 23

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 2

Dequeue 34

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 2

Dequeue:        45

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 2

Queue empty

1.Enqueue 2.Dequeue 3.Display 4.Exit
Enter choice : 4

Exiting...
```

Q2) Perform UNION and INTERSECTION set operations on singly linked lists with header node.

Solution:

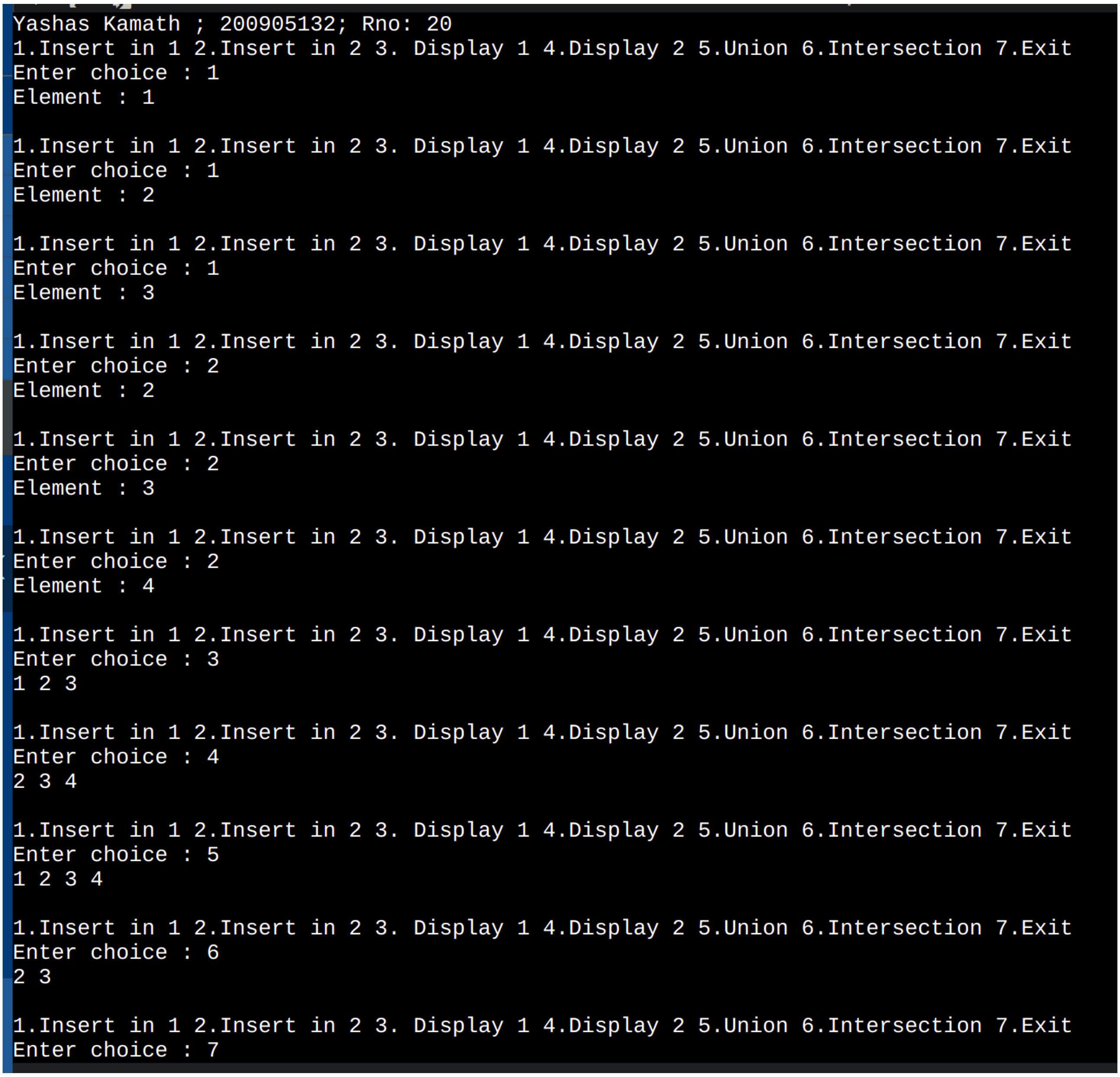
```
#include<stdio.h>
```

```
#include<stdlib.h>
typedef struct node{
int data;
struct node * next;
} * NODE;
NODE insert(NODE head,int ele){
NODE first = head->next;
NODE temp = (NODE)malloc(sizeof(struct node));
temp->data = ele;
temp->next = NULL;
if(first == NULL){
head->next = temp;
return head;
}
else if(first->next == NULL){
if(first->data > ele){
temp->next = first;
head->next = temp;
return head;
}
else if(first->data < ele){
first->next = temp;
}
else{printf("\nElement already exists in set.\n");
free(temp);
}
return head;
}
else{
NODE m = first;
while(m->next != NULL && m->next->data <= ele){
m=m->next;
}
if(m->data != ele){
temp->next = m->next;
m->next = temp;
}
else{
printf("\nElement already exists in the set.\n");
free(temp);
}
return head;
```

```
}
}
NODE UNION(NODE l1, NODE l2){
NODE uni =(NODE)malloc(sizeof(struct node));
NODE pl1 = l1->next;
NODE pl2 = l2->next;
uni->data = 0;
while(pl1 != NULL && pl2 != NULL){
if(pl1->data < pl2->data){
uni = insert(uni,pl1->data);
pl1 = pl1->next;
}
else if(pl1->data > pl2->data){
uni = insert(uni,pl2->data);
pl2 = pl2->next;
}
else{
uni = insert(uni,pl1->data);
pl1 = pl1->next;
pl2 = pl2->next;
}
}
while(pl1!=NULL){
uni = insert(uni,pl1->data);
pl1 = pl1->next;
}
while(pl2!=NULL){
uni = insert(uni,pl2->data);
pl2 = pl2->next;
}
return uni;
}NODE INTERSECTION(NODE l1, NODE l2){
NODE inter = (NODE)malloc(sizeof(struct node));
NODE pl1 = l1->next;
inter->data=0;
while(pl1!=NULL){
NODE pl2 = l2->next;
while(pl2!=NULL){
if(pl1->data == pl2->data){
inter = insert(inter,pl1->data);
break;
}
```

```
pl2=pl2->next;
}
pl1=pl1->next;
}
return inter;
}
void display(NODE head){
NODE first = head->next;
if(first == NULL){
printf("\nList empty\n");
}
else{
NODE temp = first;
while(temp->next!=NULL){
printf("%d ",temp->data);
temp=temp->next;
}
printf("%d\n",temp->data);
}
}
int main(){
    printf("Yashas Kamath ; 200905132; Rno: 20");
NODE first = (NODE)malloc(sizeof(struct node));
NODE second = (NODE)malloc(sizeof(struct node));
NODE uni = (NODE)malloc(sizeof(struct node));
NODE inter = (NODE)malloc(sizeof(struct node));
int ch,ele;
first->data = 0;
second->data = 0;
uni->data = 0;
inter->data = 0;
while(1){
printf("\n1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union
6.Intersection 7.Exit\nEnter choice : ");
scanf("%d",&ch);
switch(ch){
case 1: printf("Element : ");
scanf("%d",&ele);
first = insert(first,ele);break;
case 2: printf("Element : ");
scanf("%d",&ele);
second = insert(second,ele);
```

```
break;
case 3: display(first);
break;
case 4: display(second);
break;
case 5: uni = UNION(first,second);
display(uni);
break;
case 6: inter = INTERSECTION(first,second);
display(inter);
break;
case 7: printf("\nExiting...\n");
return 0;
}
}
}
```



```
Yashas Kamath ; 200905132; Rno: 20
1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 1

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 2

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 1
Element : 3

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 2

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 3

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 2
Element : 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 3
1 2 3

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 4
2 3 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 5
1 2 3 4

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 6
2 3

1.Insert in 1 2.Insert in 2 3. Display 1 4.Display 2 5.Union 6.Intersection 7.Exit
Enter choice : 7
```