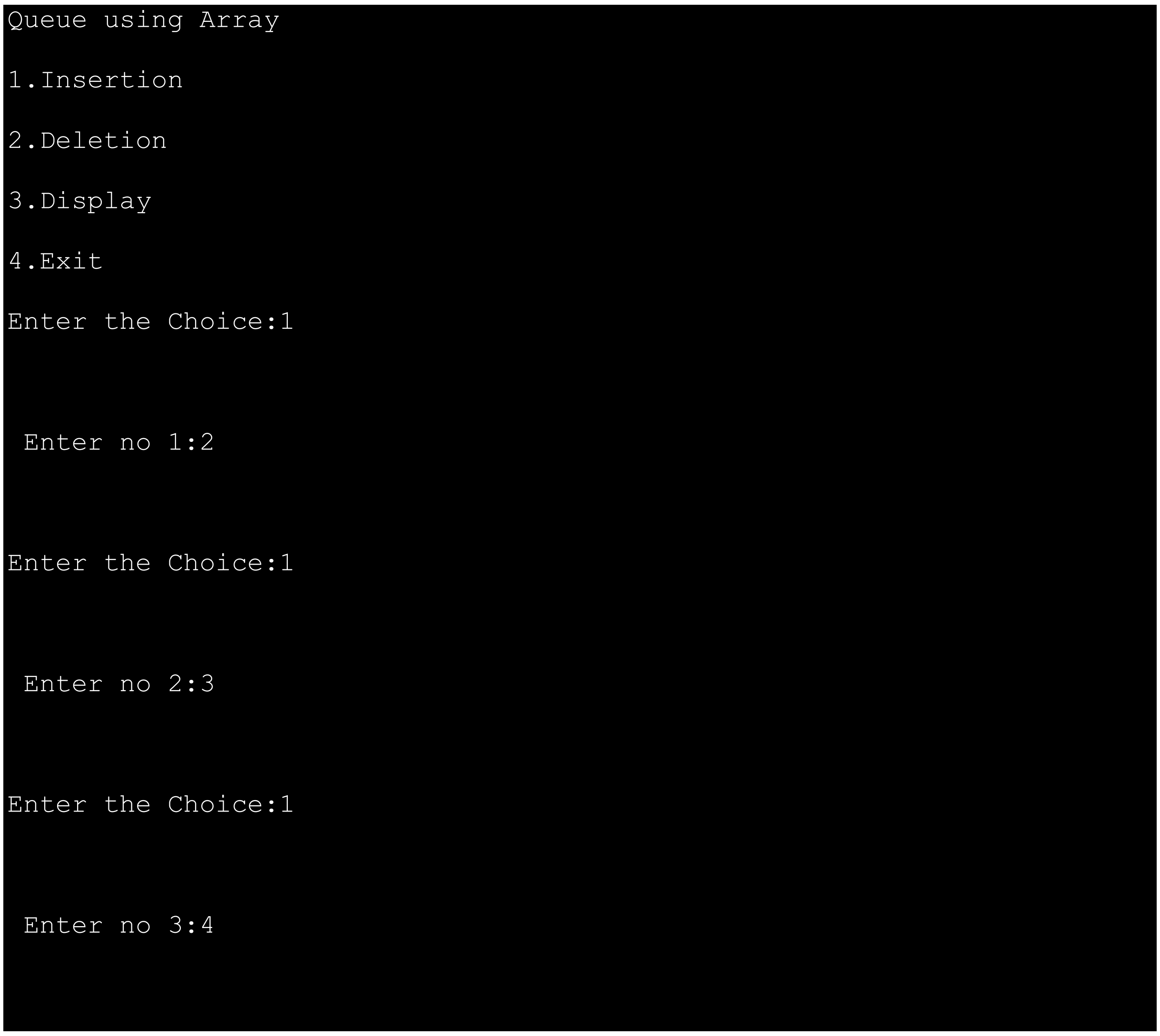
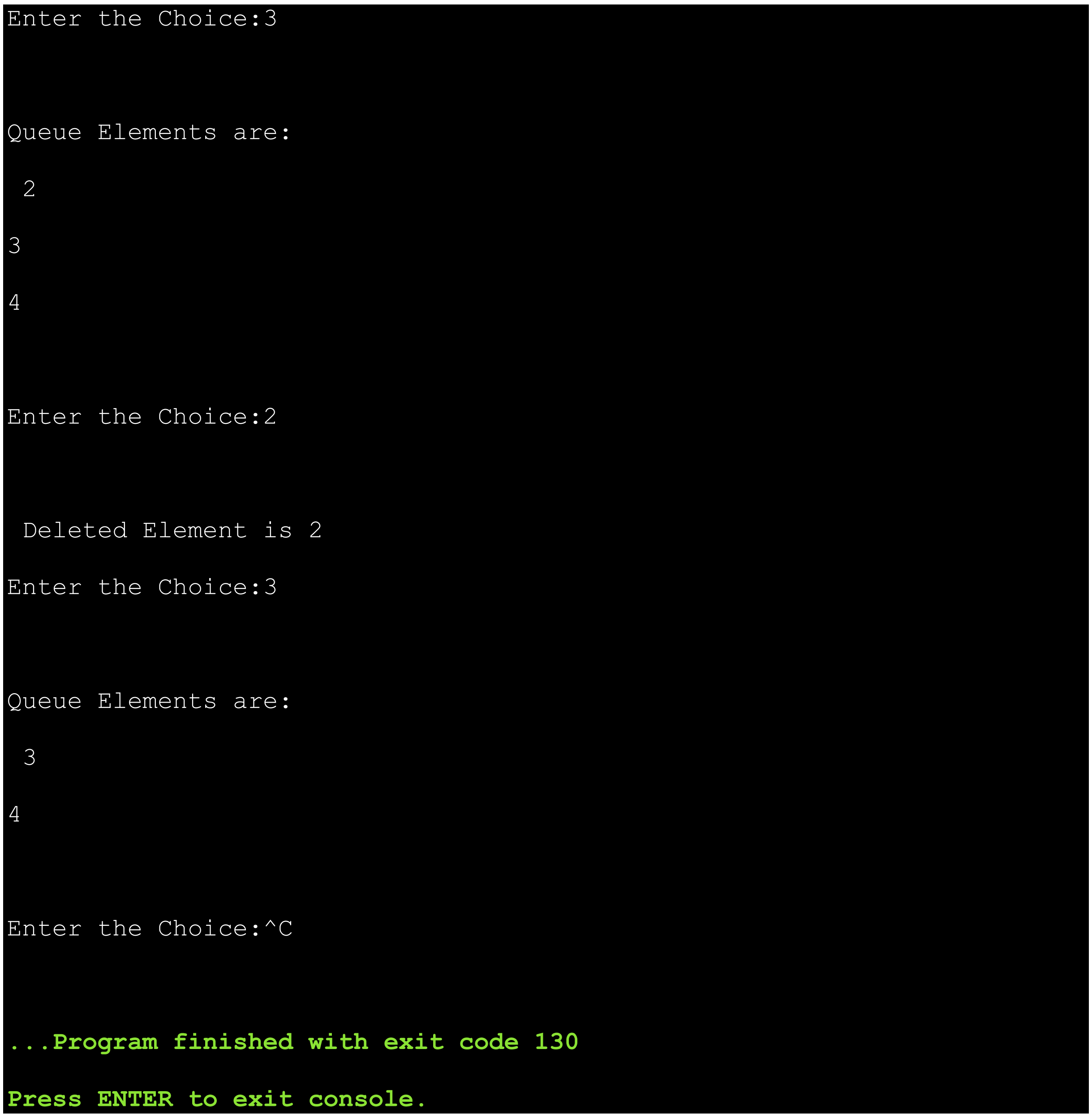
<u>QUEUE OPERATION USING ARRAY</u>

```
#include<stdio.h>
#define n 5
int main()
{
   int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;
   printf("Queue using Array");
   printf("\n1.Insertion \n2.Deletion \n3.Display \n4.Exit");
   while(ch)
   {
      printf("\nEnter the Choice:");
      scanf("%d",&ch);
      switch(ch)
      {
      case 1:
         if(rear==x)
            printf("\n Queue is Full");
         else
         {
            printf("\n Enter no %d:",j++);
            scanf("%d",&queue[rear++]);
         }
         break;
      case 2:
        if(front==rear)
        {
            printf("\n Queue is empty");
        }
        else
        {
            printf("\n Deleted Element is %d",queue[front++]);
            x++;
        }
        break;
    case 3:
        printf("\nQueue Elements are:\n ");
        if(front==rear)
            printf("\n Queue is Empty");
        else
        {
            for(i=front; i<rear; i++)
            {
                printf("%d",queue[i]);
                printf("\n");
            }
            break;
        case 4:
            exit(0);
        default:
            printf("Wrong Choice: please see the options");
          }
        }
    }
    return 0;
}
```

OUTPUT



```
Queue using Array

1.Insertion

2.Deletion

3.Display

4.Exit

Enter the Choice:1


 Enter no 1:2


Enter the Choice:1


 Enter no 2:3


Enter the Choice:1


 Enter no 3:4
```



```
Enter the Choice:3


Queue Elements are:

 2

3

4


Enter the Choice:2


 Deleted Element is 2

Enter the Choice:3


Queue Elements are:

 3

4


Enter the Choice:^C


...Program finished with exit code 130

Press ENTER to exit console.
```

LINKED LIST AND UNION OPERATION

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

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

struct node * Union(struct node * L1, struct node * L2){
	struct node * output = NULL;
	struct node * outTail = NULL;
	while(L1&&L2){
		struct node * newNode = (struct node *)
malloc(sizeof(struct node));
		newNode->next = NULL;
		if(L1->data<L2->data){
```

```
                newNode->data = L1->data;
                L1 = L1->next;
            }
            else if(L1->data>L2->data){
                newNode->data = L2->data;
                L2 = L2->next;
            }
            else{
                int data = L1->data;
                newNode->data = data;
                while(L1 && L2 && L1->data == data && L2-
>data == data){
                    L1 = L1->next;
                    L2 = L2->next;
                }
            }
            if(!output)
                output = outTail = newNode;
            else{
                outTail->next = newNode;
                outTail = outTail->next;
            }
        }
        while(L1){
```

```
		outTail->next = (struct node *) malloc(sizeof(struct
node));
		outTail = outTail->next;
		outTail->data = L1->data;
		L1 = L1->next;
	}
	while(L2){
		outTail->next = (struct node *) malloc(sizeof(struct
node));
		outTail = outTail->next;
		outTail->data = L2->data;
		L2 = L2->next;
	}
	outTail->next = NULL;
	return output;
}

struct node * intersection(struct node * L1, struct node* L2){
	if(L1 == NULL || L2 == NULL)
		return NULL;
	struct node * output = NULL;
	struct node * outTail = NULL;
	while(L1&&L2){
		if(L1->data<L2->data){
			L1 = L1->next;
```

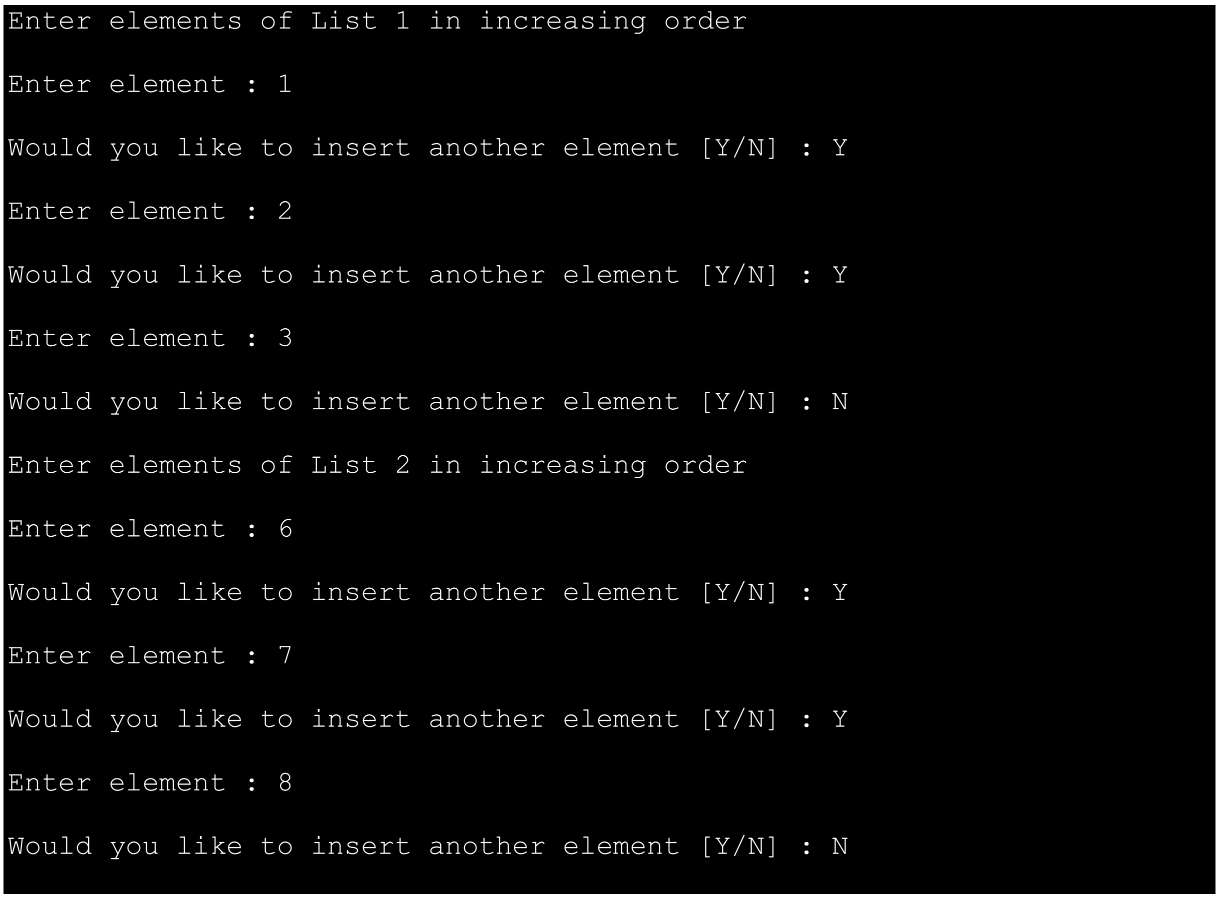
```
        }
        else if(L2->data<L1->data){
            L2 = L2->next;
        }
        else{
            int data = L1->data;
            struct node * newNode = (struct node *)
malloc(sizeof(struct node));
            newNode->data = data;
            newNode->next = NULL;
            if(output == NULL){
                outTail = output = newNode;
            }
            else{
                outTail->next = newNode;
                outTail = outTail->next;
            }
            while(L1 && L2 && L1->data == data && L2-
>data == data){
                L1 = L1->next;
                L2 = L2->next;
            }
        }
    }
    return output;
```

```
}


struct node * createList(int listNum){
        struct node * list = NULL;
        struct node * list_tail = NULL;
        printf("Enter elements of List %d in increasing
order\n",listNum);
        char ch = 'y';
        do{
                int data;
                printf("Enter element : ");
                scanf("%d",&data);
                struct node * newNode = (struct node *)
malloc(sizeof(struct node));
                newNode->data = data;
                newNode->next = NULL;
                if(list == NULL){
                        list = list_tail = newNode;
                }
                else{
                        list_tail->next = newNode;
                        list_tail = list_tail->next;
                }
                printf("Would you like to insert another element [Y/N] :
");
                scanf(" %c",&ch);
```

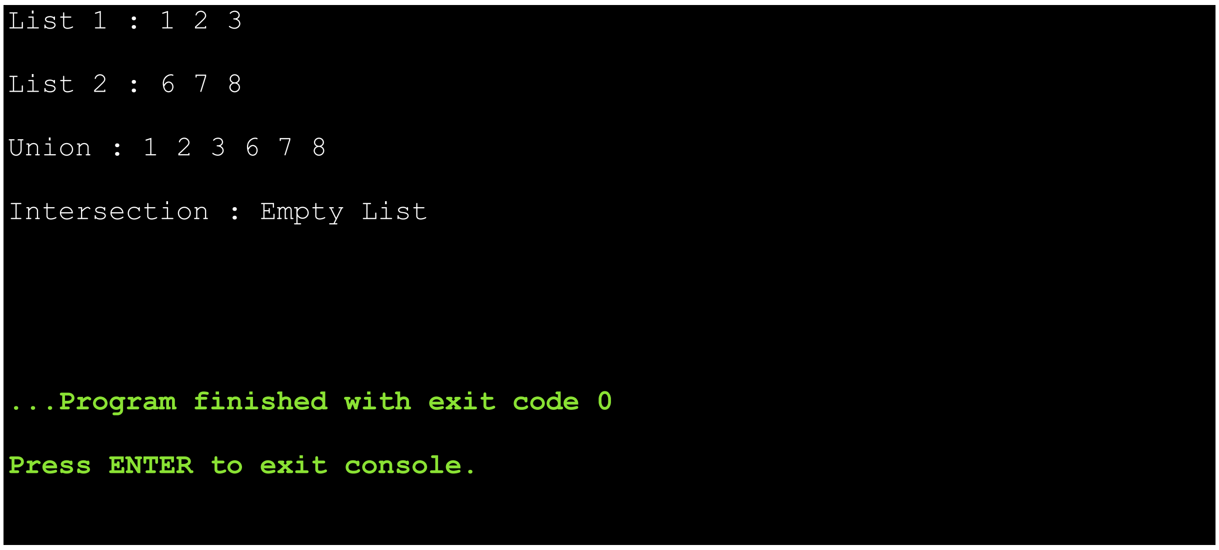
```
	}while(ch == 'y' || ch == 'Y');

	return list;
}

void print(struct node * list){
	if(list == NULL){
		printf("Empty List\n");
		return;
	}
	while(list!=NULL){
		printf("%d ",list->data);
		list = list->next;
	}
	printf("\n");
}

int main() {
	struct node * L1 = NULL;
	struct node * L2 = NULL;
	struct node * L3 = NULL;
	struct node * L4 = NULL;
	L1 = createList(1);
	L2 = createList(2);
	printf("List 1 : ");
```

```
        print(L1);
        printf("List 2 : ");
        print(L2);
        printf("Union : ");
        L3 = Union(L1, L2);
        print(L3);
        printf("Intersection : ");
        L4 = intersection(L1, L2);
        print(L4);
        return 0;
}
```

OUTPUT



```
Enter elements of List 1 in increasing order

Enter element : 1

Would you like to insert another element [Y/N] : Y

Enter element : 2

Would you like to insert another element [Y/N] : Y

Enter element : 3

Would you like to insert another element [Y/N] : N

Enter elements of List 2 in increasing order

Enter element : 6

Would you like to insert another element [Y/N] : Y

Enter element : 7

Would you like to insert another element [Y/N] : Y

Enter element : 8

Would you like to insert another element [Y/N] : N
```



```
List 1 : 1 2 3

List 2 : 6 7 8

Union : 1 2 3 6 7 8

Intersection : Empty List



...Program finished with exit code 0

Press ENTER to exit console.
```