**Module - 2**

**1. Write a C program to remove duplicate element from sorted Linked List.**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*head=NULL;

int insert (int a) {

struct node \*ptr;

ptr=(struct node\*)malloc(sizeof(struct node));

ptr->data = a;

if(head==NULL){

ptr->next=head;

head=ptr;

}

else{

struct node \*temp;

temp=head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=ptr;

ptr->next=NULL;

}

}

void remove\_repeat(){

if(head==NULL){

printf("linked list is NULL");

}

else{

struct node\* temp=head;

while(temp->next!=NULL){

if(temp->data==temp->next->data){

temp->next=temp->next->next;

}

else{

temp=temp->next;

}

}

}

}

void display(){

if(head==NULL){

printf("NO nodes present");

}

else{

struct node \*temp;

temp=head;

while(temp!=NULL){

printf("%d",temp->data);

temp=temp->next;

}

}

}

int main(){

int n,val;

scanf("%d", &n);

for(int i=0; i<n ;i++){

scanf("%d",&val);

insert(val);

}

remove\_repeat();

display();

}

**2. Write a C program to rotate a doubly linked list by N nodes.**

#include<stdio.h>

#include<stdlib.h>

struct node{

struct node \*prev;

int data;

struct node \*next;

};

struct node \*head=NULL;

int insert (int a) {

struct node \*ptr;

ptr=(struct node\*)malloc(sizeof(struct node));

ptr->data = a;

if(head==NULL){

ptr->prev=NULL;

ptr->next=head;

head=ptr;

}

else{

struct node \*temp;

temp=head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=ptr;

ptr->prev=temp;

ptr->next=NULL;

}

}

void rotate(int c){

for(int i=0;i<c;i++){

struct node \*last,\*temp;

temp=head;

last=head;

while(last->next!=NULL){

last=last->next;

}

head=temp->next;

head->prev=NULL;

temp->prev=last;

last->next=temp;

temp->next=NULL;

}

}

void display(){

if(head==NULL){

printf("NO nodes present");

}

else{

struct node \*temp;

temp=head;

while(temp!=NULL){

printf("%d\t",temp->data);

temp=temp->next;

}

}

}

int main(){

int n,val,count;

scanf("%d", &n);

printf("Enter the number of counts to be shifted : ");

scanf(" %d",&count);

for(int i=0; i<n ;i++){

scanf("%d",&val);

insert(val);

}

rotate(count);

display();

}

**3. Write a C program to sort the elements of a queue in ascending order**

#include <stdio.h>

#define size 10

int queue[size];

int first=0;

int last=0;

int swap(int \*a,int \*b){

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

}

int enqueue(int a){

if(last==0){

queue[last]=a;

last++;

}

else{

queue[last]=a;

last++;

}

}

int sort(){

for(int i=0;i<last;i++){

for(int j=last-1;j>i;j--){

if(queue[i]>queue[j]){

swap(&queue[i],&queue[j]);

}

}

}

}

int dequeue(){

if(last==0){

printf("The queue is empty");

}

else{

while(first<last){

printf("%d",queue[first]);

first++;

}

}

}

int main()

{

int n,val;

scanf("%d",&n);

for(int i=0;i<n;i++){

scanf("%d",&val);

enqueue(val);

}

sort();

dequeue();

return 0;

}

**4. List all queue function operations available for manipulation of data elements in c**

#include <stdio.h>

#define size 10

int queue[size];

int first=0;

int last=0;

int enqueue(int a){

if(last==0){

queue[last]=a;

last++;

}

else{

queue[last]=a;

last++;

}

}

void isFull(){

if(last>=max){

printf("Queue is Full");

}

else{

printf("Queue is not yet Full");

}

}

void isEmpty(){

if(last==0){

printf("Queue is Empty");

}

}

int dequeue(){

if(last==0){

printf("The queue is empty");

}

else{

while(first<last){

printf("%d",queue[first]);

first++;

}

}

}

int main()

{

int n,val,c;

scanf("%d",&n);

for(int i=0;i<n;i++){

scanf("%d",&val);

enqueue(val);

}

c=count();

printf("The total no of elements present in the queue : %d ", c);

sort();

dequeue();

return 0;

}

**5. Reverse the given string using stack**

#include <stdio.h>

#define size 100

char stack[size];

int top=-1;

void push(char c){

if(top>size){

printf("The stack is full");

}

else{

top++;

stack[top]=c;

}

}

void pop(){

if(top<0){

printf("The stack is empty");

}

else{

while(top>=0){

printf("%c",stack[top]);

top--;

}

}

}

int main()

{

int n;

char c;

scanf("%d",&n);

for(int i=0;i<n;i++){

scanf("%c",&c);

push(c);

}

pop();

return 0;

}

**6. Insert value in sorted way in a sorted doubly linked list. Given a sorted doubly linked list and a value to insert, write a function to insert the value in sorted way.**

#include <stdio.h>

#include <stdlib.h>

struct node{

struct node \*prev;

int data;

struct node \*next;

};

struct node \*head=NULL;

void insert(int val){

struct node \*ptr;

ptr=(struct node\*) malloc (sizeof(struct node));

if(head==NULL){

head=ptr;

ptr->data=val;

ptr->prev=NULL;

ptr->next=NULL;

}

else

{

struct node \*temp;

temp=head;

while(temp->next!=NULL){

temp=temp->next;

}

ptr->data=val;

temp->next=ptr;

ptr->prev=temp;

ptr->next=NULL;

}

}

void insert\_inbtw(int val){

struct node \*ptr;

ptr=(struct node\*) malloc (sizeof(struct node));

ptr->data=val;

struct node \*temp;

temp=head;

if(temp->data>val){

ptr->next=temp;

ptr->prev=NULL;

temp->prev=ptr;

head=ptr;

}

else{

while(temp!=NULL){

if(temp->data>val){

temp->prev->next=ptr;

ptr->prev=temp->prev;

ptr->next=temp;

temp->prev=ptr;

break;

}

temp=temp->next;

}

}

}

void display(){

if(head==NULL){

printf("NO nodes present");

}

else{

struct node \*temp;

temp=head;

while(temp!=NULL){

printf("%d\t",temp->data);

temp=temp->next;

}

}

}

int main(){

int n,val;

scanf("%d", &n);

for(int i=0; i<n ;i++){

scanf("%d",&val);

insert(val);

}

insert\_inbtw(9);

display();

}

**7. Write a C program to insert/delete and count the number of elements in a queue.**

#include <stdio.h>

#define size 10

int queue[size];

int first=0;

int last=0;

int enqueue(int a){

if(last==0){

queue[last]=a;

last++;

}

else{

queue[last]=a;

last++;

}

}

void isFull(){

if(last>=max){

printf("Queue is Full");

}

else{

printf("Queue is not yet Full");

}

}

void isEmpty(){

if(last==0){

printf("Queue is Empty");

}

}

int count(){

int c = last-first;

return c;

}

int dequeue(){

if(last==0){

printf("The queue is empty");

}

else{

while(first<last){

printf("%d",queue[first]);

first++;

}

}

}

int main()

{

int n,val,c;

scanf("%d",&n);

for(int i=0;i<n;i++){

scanf("%d",&val);

enqueue(val);

}

c=count();

printf("The total no of elements present in the queue : %d ", c);

sort();

dequeue();

return 0;

}

**8. Write a C program to Find whether an array is a subset of another array.**

#include<stdio.h>

int main(){

int flag=0,s;

int arr1[]={11,1,13,21,3,7};

int arr2[]={11,3,7,1};

s = sizeof(arr2)/4;

for(int i=0;i<4;i++){

for(int j=0;j<6;j++){

if(arr2[i]==arr1[j]){

flag++;

}

}

}

if(flag==s){

printf("arr2 is a subset of arr1");

}

else{

printf("arr2 is not a subset of arr1");

}

}