//RIGHT

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#include<ctype.h>

#define SIZE 50

typedef struct node

{

int seats;

float per\_votes;;

struct node \*left , \*right;

int r1,r2;

}node;

typedef struct Qtree

{

node \*root;

}Qtree;

void displayseats(node \*qt,char names [][SIZE])

{

node \*p;

p=qt;

if(qt!=NULL)

{

displayseats(p->left,names);

printf("\nSeats: %d\tInRegion :\t%s---%s",p->seats,names[p->r1],names[p->r2]);

displayseats(p->right,names);

}

}

void displayvotes(node \*qt,char names [][SIZE])

{

node \*p;

p=qt;

if(qt!=NULL)

{

displayvotes(p->left,names);

printf("\nPercent Votes : %f\tRegion :\t%s---%s",p->per\_votes,names[p->r1],names[p->r2]);

displayvotes(p->right,names);

}

}

node \*create(node \*qt,int seats\_distr[],float percent\_votes[],int start,int end)

{

int mid;

node \*p;

p=(node\*)malloc(sizeof(node));

if(start==end)

{

p->seats=seats\_distr[start];

p->per\_votes=percent\_votes[start];

p->left=NULL;

p->right=NULL;

p->r1=p->r2=start;

qt=p;

return qt;

}

else if(start<end)

{

mid=(start+end)/2;

p->left=create(qt,seats\_distr,percent\_votes,start,mid);

p->left->r1=start;

p->left->r2=mid;

p->right=create(qt,seats\_distr,percent\_votes,mid+1,end);

p->right->r1=mid+1;

p->right->r2=end;

p->seats=p->left->seats+p->right->seats;

p->per\_votes=(p->left->per\_votes+p->right->per\_votes)/2;

p->r1=0;

p->r2=end;

qt=p;

return qt;

}

}

int query\_seats(node \*qt,int qs,int qe)

{

int y,x1,x2;

node \*p;

p=qt;

if(p!=NULL)

{

if(p->r1>=qs && p->r2<=qe)

{

y=p->seats;

return y;

}

x1=query\_seats(qt->left,qs,qe);

x2=query\_seats(qt->right,qs,qe);

return (x1+x2);

}

}

int query\_winseats(node \*qt,int qs,int qe)

{

int y,x1,x2;

node \*p;

p=qt;

if(p!=NULL)

{

if(p->r1>=qs && p->r2<=qe)

{

y=(p->seats\*(p->per\_votes))/100;

return y;

}

x1=query\_winseats(qt->left,qs,qe);

x2=query\_winseats(qt->right,qs,qe);

return (x1+x2);

}

}

node \*update\_lazy\_roots(node \*qt,int qs,int qe,int update\_val)

{

node \*p;

p=qt;

if(p!=NULL)

{

if(p->r1>=qs && p->r2<=qe && p->r1==p->r2)

{

p->seats+=update\_val;

}

qt->left=update\_lazy\_roots(qt->left,qs,qe,update\_val);

qt->right=update\_lazy\_roots(qt->right,qs,qe,update\_val);

return qt;

}

}

node \*update\_lazy\_tree(node \*qt,int qs,int qe)

{

node \*p;

p=qt;

if(p!=NULL)

{

p->left=update\_lazy\_tree(p->left,qs,qe);

p->right=update\_lazy\_tree(p->right,qs,qe);

if((p->r1<=qs || p->r2>=qe) && p->r1<p->r2)

{

p->seats=p->left->seats + p->right->seats;

}

qt=p;

return qt;

}

}

int main()

{

Qtree qt;

qt.root=NULL;

int seats\_distr[SIZE],n,i,x,y,rl,rr,update\_val;

float percent\_votes[SIZE];

char names[SIZE][SIZE];

char elec\_party[SIZE];

printf("\t\*\*\*\*\*\*\*\*\*\*\*\* 'WELCOME TO ELECTION COMMISSION' \*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("\nEnter Name Of Election Party : ");

scanf("%s",elec\_party);

printf("\nEnter Number Of Regions Elections Held : ");

scanf("%d",&n);

printf("\nEnter Names Of Regions : ");

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

{

printf("\nEnter Name Of Region [%d] : ",i+1);

scanf("%s",names[i]);

}

printf("\nEnter The Region's Seats: ");

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

{

printf("\nEnter Total Seats Of Region [%s] : ",names[i]);

scanf("%d",&seats\_distr[i]);

printf("\nEnter Percentage Votes For Party : %s In Region [%s] : ",elec\_party,names[i]);

scanf("%f",&percent\_votes[i]);

}

qt.root=create(qt.root,seats\_distr,percent\_votes,0,n-1);

printf("\nSeats Distribution In The %d Regions are : \n",n);

displayseats(qt.root,names);

printf("\n");

printf("\n");

printf("\nPercentage Votes Distribution In The %d Regions are : \n",n);

displayvotes(qt.root,names);

printf("\n\nEnter The Range Of Regions For Query Of Seats : ");

printf("\nEnter Start Region : ");

scanf("%d",&rl);

printf("\nEnter End Region : ");

scanf("%d",&rr);

x=query\_seats(qt.root,rl-1,rr-1);

printf("\nTotal Seats For Region [%s,%s] are : %d",names[rl-1],names[rr-1],x);

printf("\n\nEnter The Range Of Regions For Query Of Seats Won By Party %s : ",elec\_party);

printf("\nEnter Start Region : ");

scanf("%d",&rl);

printf("\nEnter End Region : ");

scanf("%d",&rr);

x=query\_winseats(qt.root,rl-1,rr-1);

printf("\nTotal Seats For Region [%s,%s] are : %d",names[rl-1],names[rr-1],x);

printf("\nEnter Value To Update : ");

scanf("%d",&update\_val);

printf("\n");

printf("\n\nEnter The Range Of Regions For Update Of Seats : %s : ",elec\_party);

printf("\nEnter Start Region : ");

scanf("%d",&rl);

printf("\nEnter End Region : ");

scanf("%d",&rr);

qt.root=update\_lazy\_roots(qt.root,rl-1,rr-1,update\_val);

qt.root=update\_lazy\_tree(qt.root,rl-1,rr-1);

printf("\nSeats Distribution In The %d Regions are : \n",n);

displayseats(qt.root,names);

printf("\nPercentage Votes Distribution In The %d Regions are : \n",n);

displayvotes(qt.root,names);

return 0;

}