Topic : IQ Level Detector

Group Members :

1) Dishita Ashar(3)

2) Vaibhavi Chincholkar(12)

3) Mansi Dwivedi(16)

Code in text format :

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

/////////////////////////////////////////////////////////////////// for hash table //////////////////////////////////////////////////////////////////////

int n;

int j=0;

int pos=1;

int tempo;

struct node //defining structure for the hash table node

{

int data;

struct node\* next;

char \*q; //for storing the question

char ans; //for storing the right option character

char \*ans1; //for storing option 1

char \*ans2; //for storing option 2

char \*ans3; //for storing option 3

char \*ans4; //for storing option 4

};

void insert(struct node \*ar[n],int val,char \*q1,char \*ans1,char \*ans2,char \*ans3,char \*ans4,char ans) //function to create hash table

{

int len1,len2,len3,len4,len5;

len1=strlen(q1);

len2=strlen(ans1);

len3=strlen(ans2);

len4=strlen(ans3);

len5=strlen(ans4);

struct node \*new1;

new1=(struct node \*)malloc(sizeof(struct node)); //allocating memory

new1->data=val;

new1->next=NULL;

new1->q=(char\*)malloc(len1\*sizeof(char));

strcpy(new1->q,q1);

new1->ans1=(char\*)malloc(len2\*sizeof(char));

strcpy(new1->ans1,ans1);

new1->ans2=(char\*)malloc(len3\*sizeof(char));

strcpy(new1->ans2,ans2);

new1->ans3=(char\*)malloc(len4\*sizeof(char));

strcpy(new1->ans3,ans3);

new1->ans4=(char\*)malloc(len5\*sizeof(char));

strcpy(new1->ans4,ans4);

new1->ans=ans;

int key;

key=val%n; //Finding key using function

if(ar[key]==NULL) //If empty add number

{

ar[key]=new1;

}

else

{

struct node \*t1=ar[key];

while(t1->next!=NULL)

{

t1=t1->next;

}

t1->next=new1;

}

}

struct node \*search(struct node \*ar[n],int snum) //searching a number in the hash table

{

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

{

struct node \*t=ar[i];

while(t!=NULL)

{

if(t->data==snum) return(t); //If number is found return it

else t=t->next; //else update pointers

}

}

}

/////////////////////////////////////////////////////////////////// for avl //////////////////////////////////////////////////////////////////////

struct avl //structure definition for the avl tree

{

int data;

int bal;

int level;

struct avl \*par;

struct avl \*left;

struct avl \*right;

}\*root;

struct avl \*tempb=NULL;

struct avl \*create(struct avl \*parent,int data) //function for creating a new node and also assigning it's parents

{

struct avl \*temp;

temp=(struct avl\*)malloc(sizeof(struct avl));

temp->data=data;

temp->par=parent;

temp->bal=0;

temp->left=temp->right=NULL;

return temp;

}

int max(int m,int n) //function to find the maximum

{

return(m>n)?m:n;

}

int hei(struct avl \*node) //function to find the height of a given node

{

if(node==NULL)return 0;

return(1+max(hei(node->left),hei(node->right)));

}

struct avl \*balance(struct avl \*root,int data) //function for assigning the balance to nodes

{

if(root==NULL)return NULL;

root->bal=hei(root->left)-hei(root->right);

if(root->data>data)balance(root->left,data);

else if(root->data<data)balance(root->right,data);

else

{

return root;

}

}

struct avl \*check(struct avl \*root) //function for checking whether the node is unbalanced

{

if(root==NULL)return NULL;

if(root->bal==1 || root->bal==-1 || root->bal==0)check(root->par);

else return root;

}

struct avl \*rightr(struct avl \*oldp) //function to perform right rotation(LL)

{

struct avl \*newp = oldp->left;

struct avl \*orp = newp->right;

newp->right = oldp;

newp->par=oldp->par;

if(oldp->par!=NULL && oldp->par->data > newp->data)oldp->par->left=newp;

else if(oldp->par!=NULL && oldp->par->data < newp->data)oldp->right=newp;

oldp->par=newp;

oldp->left = orp;

if(orp!=NULL)orp->par=oldp;

return newp;

}

struct avl \*leftr(struct avl \*oldp) //function to perform left rotation(RR)

{

struct avl \*newp = oldp->right;

struct avl \*orp = newp->left;

newp->left = oldp;

newp->par=oldp->par;

if(oldp->par!=NULL && oldp->par->data < newp->data)oldp->par->right=newp;

else if(oldp->par!=NULL && oldp->par->data > newp->data)oldp->par->left=newp;

oldp->par=newp;

oldp->right = orp;

if(orp!=NULL)orp->par=oldp;

return newp;

}

struct avl \*leftrig(struct avl \*oldp) //function to perform left and then right rotation(LR)

{

struct avl \*beechp=oldp->left;

struct avl \*newp=beechp->right;

struct avl \*orpl=newp->left;

struct avl \*orpr=newp->right;

newp->par=oldp->par;

if(oldp->par!=NULL)

{

if(oldp->par->data < oldp->data)oldp->par->right=newp;

else oldp->par->left=newp;

}

newp->left=beechp;

beechp->par=newp;

beechp->right=orpl;

if(orpl!=NULL)orpl->par=beechp;

newp->right=oldp;

oldp->par=newp;

oldp->left=orpr;

if(orpr!=NULL)orpr->par=oldp;

return newp;

}

struct avl \*rigleft(struct avl \*oldp) //function to perform right and then left rotation(RL)

{

struct avl \*beechp=oldp->right;

struct avl \*newp=beechp->left;

struct avl \*orpr=newp->right;

struct avl \*orpl=newp->left;

newp->par=oldp->par;

if(oldp->par!=NULL)

{

if(oldp->par->data > oldp->data)oldp->par->left=newp;

else oldp->par->right=newp;

}

newp->right=beechp;

beechp->par=newp;

beechp->left=orpr;

if(orpr!=NULL)orpr->par=beechp;

newp->left=oldp;

oldp->par=newp;

oldp->right=orpl;

if(orpl!=NULL)orpl->par=oldp;

return newp;

}

struct avl \*insert1(struct avl \*temp1,int data) //function for the insertion of a given node

{

if(temp1==NULL)return(create(tempb,data));

if(data>temp1->data)

{

tempb=temp1;

temp1->right=insert1(temp1->right,data);

}

else

{

tempb=temp1;

temp1->left=insert1(temp1->left,data);

}

return(temp1);

}

struct avl \*height(struct avl \*node) //function for calculating the longest height subtree

{

int a=hei(node->left);

int b=hei(node->right);

return(a>b?node->left:node->right);

}

int gethei(struct avl \*node) //function for actually getting the height in int format

{

int a=hei(node->left);

int b=hei(node->right);

return(a>b?a:b);

}

int y;

void giveheight(struct avl \*rooti) //function for assigning the given heights to each node

{

if(rooti!=NULL)

{

rooti->level=gethei(rooti);

giveheight(rooti->left);

giveheight(rooti->right);

}

}

//////////////////////////////////////////////////////////////////// for bst //////////////////////////////////////////////////////////////////////

struct bst //structure for creating a bst required to store the scores of all players

{

int data; //his/her score

char person[30]; //player's username

struct bst \*par;

struct bst \*left;

struct bst \*right;

}\*root1;

struct bst \*tempa=NULL;

struct bst \*createbst(struct bst \*parent,int data,char name[]) //function for creating the bst node

{

struct bst \*temp;

temp=(struct bst\*)malloc(sizeof(struct bst));

temp->data=data;

temp->par=parent;

temp->left=temp->right=NULL;

strcpy(temp->person,name);

return temp;

}

struct bst \*insert2(struct bst \*temp1,int data,char name[]) //function for creating the tree and entering corresponding values.

{

if(temp1==NULL)return(createbst(tempa,data,name));

if(data>temp1->data)

{

tempa=temp1;

temp1->right=insert2(temp1->right,data,name);

}

else if(data<temp1->data)

{

tempa=temp1;

temp1->left=insert2(temp1->left,data,name);

}

else

{

tempa=temp1;

temp1->left=insert2(temp1->left,data,name);

}

return(temp1);

}

void inorderbst(struct bst \*root) //function for printing the final scores

{

if (root != NULL)

{

inorderbst(root->right);

if(tempo==root->data && pos!=1)

{

pos--;

}

else

{

tempo=root->data;

}

printf("\t%s\t|\t%d/130\t|\t%d\t\t|\n",root->person,root->data+1,pos++);

inorderbst(root->left);

}

}

/////////////////////////////////////////////////////////////////// for play game //////////////////////////////////////////////////////////////////////

int play(struct avl \*root,int score,struct node \*ar[n]) //function for actual game implementation

{

struct node \*temp;

if(root==NULL) return score;

else

{

int i=root->data;

temp=search(ar,i);

char ansa[20];

printf("\n%d) %s\n",y,temp->q);

printf("A) %s B) %s C) %s D) %s\n",temp->ans1,temp->ans2,temp->ans3,temp->ans4);

y++;

printf("\n");

scanf("%s",ansa);

//Various test cases

if(strlen(ansa)>1 || strlen(ansa)==0) //if the input entered by the user is more than 1 character then user should try again

{

printf("Invalid Input!\nTry Again\n");

y--;

}

else

{

int result=ansa[0]-temp->ans;

if((ansa[0]==temp->ans) || result==32 || result==-32) //if the user entered the correct option but just that it could be upper or lower case

{

score=score+root->data;

root=root->right;

printf("Correct!\n");

}

else if((ansa[0]>=65 && ansa[0]<=68) || (ansa[0]>=97 && ansa[0]<=100)) //if the user entered any other alphabetical character out of the options

{

score=score-root->data;

root=root->left;

printf("Incorrect!\n");

}

else //if user entered any other character

{

printf("Invalid Input!\nTry Again\n");

y--;

}

}

play(root,score,ar);

}

}

int main()

{

/////////////////////////////////////////////////////////////////// for hash map storing //////////////////////////////////////////////////////////////////////

n=13;

struct node \*ar[n];

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

{

ar[i]=NULL;

}

char A='A';

char B='B';

char C='C';

char D='D';

insert(ar,1,"Total how many colours are there in a rainbow?","7","8","9","6",A);

insert(ar,2,"What is 7 squared?","47","49","50","45",B);

insert(ar,3,"What temperature centigrade does water boil at?","98","97","0","100",D);

insert(ar,4,"What number is a dozen?","12","10","6","11",A);

insert(ar,5,"What is 7 cubed?","443","433","334","343",D);

insert(ar,6,"What is the biggest planet in our solar system?","Saturn","Uranus","Jupiter","Earth",C);

insert(ar,7,"What is taller, an elephant or a giraffe?","Elephant","Both are equal","Giraffe","No Answer",C);

insert(ar,8,"C programs are converted into machine language with the help of a \_\_\_?","Compiler","Terminal","CodeConverter","None",A);

insert(ar,9,"Name of the wife of the Pandavas in the Indian epic Mahabharat:","Sati","Draupadi","Kunti","Gandhari",B);

insert(ar,10,"What is 3 factorial?","6","5","3","0",A);

insert(ar,11,"What part of a plant conducts photosynthesis?","Fruits","Leaf","Flower","Stem",B);

insert(ar,12,"How many notes are there in a musical scale?","8","9","7","6",C);

insert(ar,13,"What kind of energy is contained in a rock on a cliff?","Kinetic","Potential","Both","None",B);

insert(ar,14,"What is the smallest planet in our solar system?","Venus","Mars","Earth","Mercury",D);

insert(ar,15,"Elon Musk is the CEO of which global brand.","Tesla","BMW","Mercedes","None",A);

insert(ar,16,"How many years are there in a millennium?","1000","10","100","10000",A);

insert(ar,17,"Which operating system does a Google Pixel phone use?","Mac","Windows","Android","None",C);

insert(ar,18,"One KB has how many bytes?","1000","512","256","1024",D);

insert(ar,19,"What number is a baker's dozen?","12","10","14","13",D);

insert(ar,20,"Where were the 2020 Olympics supposed to be held?","Tokyo","Sparta","Beijing","Mumbai",A);

insert(ar,21,"The short form for Chinese army is ?","CRA","PLA","PLB","PRA",B);

insert(ar,22,"How many permanent members are there on the UN security council?","7","10","5","6",C);

insert(ar,23,"What is the capital city of Australia?","Canberra","Sydney","Melbourne","Brisbane",A);

insert(ar,24,"Which Mughal emperor was a good Veena player?","Babur","ShahJahan","Aurangzeb","Akbar",C);

insert(ar,25,"If you have cryophobia, what are you afraid of?","Chocolates","Sleep","Cold","Ice",D);

insert(ar,26,"Name the nuclear plant that closed forever in Ukraine.","Fukushima","Chernobyl","Hanul","Bruce",B);

insert(ar,27,"What do you call a stock market trend that is neither bullish nor bearish?","Chicken Market","Puppy Market","Lion Market","Robin Market",A);

insert(ar,28,"What was the name of the first Space Shuttle?","Apollo","Chandrayan","Columbia","None",C);

insert(ar,29,"How many women have won the Nobel Prize for Economics since 1969?","0","1","2","3",A);

insert(ar,30,"'Triple Therapy drug cocktail' is a medicine to which disease?","Rabies","Corona","Polio","AIDS",D);

insert(ar,31,"Which Disney Princess has the least amount of screen time?","Aurora","Tiana","Jasmine","Tinker",A);

/////////////////////////////////////////////////////////////////// for avl tree //////////////////////////////////////////////////////////////////////

struct avl \*rootb,\*now,\*crit,\*l;

rootb=NULL;

int i=1;

root=create(rootb,i);

i=i+1;

while(i<=31)

{

root=insert1(root,i);

now=balance(root,i);

crit=check(now);

if(crit!=NULL)

{

if(crit->bal > 1 && i< crit->left->data)l=rightr(crit);

else if(crit->bal < -1 && i > crit->right->data)l=leftr(crit);

else if(crit->bal > 1 && i > crit->left->data)l=leftrig(crit);

else l=rigleft(crit);

if(l->par==NULL)root=l;

i=i+1;

}

else

{

i++;

continue;

}

}

giveheight(root);

/////////////////////////////////////////////////////////////////// for bst making //////////////////////////////////////////////////////////////////////

struct bst \*roota;

roota=NULL;

/////////////////////////////////////////////////////////////////// for actual game //////////////////////////////////////////////////////////////////////

printf("\n HELLO! \n");

int score;

int p=0;

int dumb=-11;

int o=1;

int no;

char name[30];

printf("\n Welcome to the IQ detector test! \n");

printf("\nEnter the number of players : ");

scanf("%d",&no);

printf("\n Let's begin! \n");

printf("-------------------------------------------------------------------------------\n");

printf("\nEnter just the option character!\n");

while(no>=o)

{

y=1;

score=0;

printf("\n Enter User-name : ");

scanf("%s",&name);

printf("\n...............................................................................\n");

printf(" Player %d \n",o);

printf("...............................................................................\n");

score=play(root,score,ar);

if(o==1) root1=createbst(roota,score,name);

else root1=insert2(root1,score,name);

printf("\n////////////////////////////////////////////////////////////////////////////////\n\n");

o++;

}

printf("\n------------------------------ Score Board ------------------------------------\n");

printf("\n---------------------------------------------------------\n");

printf("User-Name\t|\tScore\t|\tPosition\t|\n");

printf("\n---------------------------------------------------------\n");

tempo=root1->data;

inorderbst(root1);

printf("---------------------------------------------------------\n");

return 0;

}

Output :





