#include <iostream>

#include <string>

using namespace std;

struct leaf //a single node of the tree

{

int data;

leaf\* left = NULL;

leaf\* right = NULL;

};

struct Nodemath{ //a single node of the math linked list

string qt;

string ans;

Nodemath\* next = NULL;

};

class list{ //the linked list itself

public:

Nodemath\* head = new Nodemath;

Nodemath\* point;

createEasy() //level 1 for maths test

{

head->qt = "Easy: What is 1 + 1 ? (answer in integer or fraction)";

head->ans = "2";

point = new Nodemath;

head->next = point;

point->qt = "Easy: What is 2 + 1 ? (answer in integer or fraction)";

point->ans = "3";

point->next = new Nodemath;

point = point->next;

point->qt = "Easy: What is 2 + 2 ?(answer in integer or fraction)";

point->ans = "4";

point->next = new Nodemath;

point = point->next;

point->qt = "Easy: What is 3 + 2 ? (answer in integer or fraction)";

point->ans = "5";

point->next = new Nodemath;

point = point->next;

point->qt = "Easy: What is 3 + 3 ? (answer in integer or fraction)";

point->ans = "6";

point->next = NULL;

point = point->next;

//displayList();

cout<<endl; //for checking purposes only

}

createMed() //level 2 for maths test

{

head->qt = "Med: What is 5% of Rs.100 ? (answer in integer or fraction)";

head->ans = "5";

point = new Nodemath;

head->next = point;

point->qt = "Med: What is 1 + 4 / 2 \* 5 ? (answer in integer or fraction)";

point->ans = "11";

point->next = new Nodemath;

point = point->next;

point->qt = "Med: What is the ratio in the series 5, 15, 45, 135 (answer in integer or fraction)";

point->ans = "3";

point->next = new Nodemath;

point = point->next;

point->qt = "Med: Two angles of a triangle are 90 and 30. What is the third? (answer in integer or fraction)";

point->ans = "60";

point->next = new Nodemath;

point = point->next;

point->qt = "Med: If two lines are perpendicular to each other the angle between then is? (answer in integer or fraction)";

point->ans = "90";

point->next = NULL;

point = point->next;

//displayList();cout<<endl; //for checking purposes only

}

createHard() //level 3 for maths test

{

head->qt = "Hard: Solve for x-> x+y=5, x-y=3 (answer in integer or fraction)";

head->ans = "4";

point = new Nodemath;

head->next = point;

point->qt = "Hard: The ratio in the following series is ? 15, 45, 135, 405... (answer in integer or fraction)";

point->ans = "3";

point->next = new Nodemath;

point = point->next;

point->qt = "Hard: If the square of x is y and x is half of y, y is ? (for y>0)? (answer in integer or fraction)";

point->ans = "4";

point->next = new Nodemath;

point = point->next;

point->qt = "Hard: If Sara is half as old as her mother and next year her mother will be 25 years older than her, right now Sara is ? (answer in integer or fraction) ";

point->ans = "25";

point->next = new Nodemath;

point = point->next;

point->qt = "Hard: The mean of the following data set is ? : 1, 4, 6, 7, 8, 10 (answer in integer or fraction)";

point->ans = "6";

point->next = NULL;

point = point->next;

//displayList();cout<<endl; //for checking purposes only

}

createEasy1() //level 1 for english test

{

head->qt = "Easy: What is the opposite of the word enter?";

head->ans = "exit";

point = new Nodemath;

head->next = point;

point->qt = "Easy: What is opposite of the word close?";

point->ans = "open";

point->next = new Nodemath;

point = point->next;

point->qt = "Easy: What is the opposite of the word toddler?";

point->ans = "adult";

point->next = new Nodemath;

point = point->next;

point->qt = "Easy: The word wander means to walk slowly and aimlessly (True/False)?";

point->ans = "true";

point->next = new Nodemath;

point = point->next;

point->qt = "Easy: the word fragile means strong (true/false)?";

point->ans = "false";

point->next = NULL;

point = point->next;

//displayList();cout<<endl; //for checking purposes only

}

createMed1() //level 2 for english test

{

head->qt = "Med: She could not go either by bus \_\_\_\_\_\_ by train. \na)no\n b)or\n c)nor\n d)and\n";

head->ans = "or";

point = new Nodemath;

head->next = point;

point->qt = "Med: He \_\_\_\_ take the exam next year.";

point->ans = "will";

point->next = new Nodemath;

point = point->next;

point->qt = "Med: As you \_\_\_, so shall you reap";

point->ans = "sow";

point->next = new Nodemath;

point = point->next;

point->qt = "Med: Don't laugh \_\_\_ the beggars.'";

point->ans = "at";

point->next = new Nodemath;

point = point->next;

point->qt = "Med: could is the past form of \_\_\_";

point->ans = "can";

point->next = NULL;

point = point->next;

//displayList();cout<<endl; //for checking purposes only

}

createHard1() //level 3 for english test

{

head->qt = "Hard: 'The green-eyed monster' means:\na)hatred\nb)love\nc)life\nd)jealousy";

head->ans = "jealousy";

point = new Nodemath;

head->next = point;

point->qt = "Hard: Retype the correct spelling: \na)affedevit\nb)affidavit\nc)afidevit\nd)affidevit";

point->ans = "affidavit";

point->next = new Nodemath;

point = point->next;

point->qt = "Hard: A prepositional phrase consists of a preposition and its object. (True/False)?";

point->ans = "false";

point->next = new Nodemath;

point = point->next;

point->qt = "Hard: Phrases are grammatical units that consist of one or more words. (True/False)? ";

point->ans = "true";

point->next = new Nodemath;

point = point->next;

point->qt = "Hard: 'The national anthem is being sung by Jason this time.' \nThe above sentence is Active or Passive?";

point->ans = "passive";

point->next = NULL;

point = point->next;

//displayList();cout<<endl; //for checking purposes only\*/

}

void displayList()

{

Nodemath\* ptrr = head;

while(ptrr != NULL)

{

cout<<ptrr->qt<<endl;

ptrr = ptrr->next;

}

}

};

class Tree //the tree itself

{

public:

leaf\* root = NULL;

Tree() //constructor to initialize a tree with the below mentioned values

{

insert(15);

insert(8);

insert(23);

insert(4);

insert(11);

insert(19);

insert(27);

insert(2);

insert(5);

insert(9);

insert(13);

insert(17);

insert(21);

insert(25);

insert(29);

insert(1);

insert(3);

insert(6);

insert(7);

insert(8);

insert(10);

insert(12);

insert(14);

insert(16);

insert(18);

insert(20);

insert(22);

insert(24);

insert(26);

insert(28);

insert(30);

//displayT();cout<<endl; //for checking purposes only

root->data = 1;

filler(root);

}

void filler(leaf\* fill) //function to change values of tree to level 1,2,3

{

if(fill->left == NULL)

{

return;

}

else{

switch(fill->data)

{

case 1:{

fill->left->data = 1;

fill->right->data = 2;

break;

}

case 2:{

fill->left->data = 1;

fill->right->data = 3;

break;

}

case 3:{

fill->left->data = 2;

fill->right->data = 3;

break;

}

}

filler(fill->left);

filler(fill->right);}

}

void insert(int d)

{

leaf\* ptr = root;

leaf\* prev = NULL;

if(root == NULL)

{

root = new leaf;

root->data = d;

return;

}

while(ptr != NULL)

{

prev = ptr;

if(d < ptr->data)

{

ptr = ptr->left;

}

else

{

ptr = ptr->right;

}

}

if(d<prev->data)

{

prev->left = new leaf;

prev->left->data = d;

}

else

{

prev->right = new leaf;

prev->right->data = d;

}

}

void displayT()

{

displayTree(root);

return;

}

void displayTree(leaf \*ptr)

{

if(ptr==NULL)

return;

displayTree(ptr->left);

cout<<ptr->data<<" ";

displayTree(ptr->right);

} /\*PROOF THAT THE TREE COULD NOT BE CREATED THROUGH RECURSION :(

void createTree(leaf\* ptr)

{

ptr->left = new leaf;

ptr->right = new leaf;

if(ptr->data == 1)

{

ptr->left->data = 1;

ptr->right->data = 2;

}

else

{

if(ptr->data == 2)

{

ptr->left->data = 1;

ptr->right->data = 3;

}

else

{

if(ptr->data ==3)

{

ptr->left->data = 2;

ptr->right->data = 3;

}

}

}

while(count < 3 )

{

count++;

createTree(ptr->left);

createTree(ptr->right);

}

}

\*/

};

int main()

{

Tree single; //tree class object

list Easy, Med, Hard; //linked list object

string A;

leaf\* Pl = single.root;

int round = 0;

int marks = 0;

Nodemath\* Pe = Easy.head;

Nodemath\* Pm = Med.head;

Nodemath\* Ph = Hard.head;

string ansr = "";

cout<<"\t\t\t\t\tPROGRESS BASED TESTING PROGRAM";

cout<<"\n\nDo you want to take the English or Math Test?\n"; //giving user options to select between two subjects

cin>>A;

if (A == "math"){

Easy.createEasy(); cout<<endl;

Med.createMed(); cout<<endl;

Hard.createHard(); cout<<endl;

}

else if(A == "english"){

Easy.createEasy1(); cout<<endl;

Med.createMed1(); cout<<endl;

Hard.createHard1(); cout<<endl;

}

for(round = 0; round<5; round++){ //loop for running the code five times for five questions

switch(Pl->data){

case 1:{

cout<<Pe->qt<<endl;

cin>>ansr;

if(ansr == Pe->ans)

{

Pl = Pl->right; //if the user answers correctly, he/she is levelled up

marks = marks + 1;

cout<<endl;

cout<<"Correct!"<<endl;

cout<<endl;

}

else

{

Pl = Pl->left; //if the user answers incorrectly, he/she is levelled down

cout<<endl;

cout<<"Wrong!"<<endl;

cout<<endl;

}

Pe = Pe->next;

break;

}

case 2:{

cout<<Pm->qt<<endl;

cin>>ansr;

if(ansr == Pm->ans)

{

Pl = Pl->right;

marks = marks + 2;

cout<<endl;

cout<<"Correct!"<<endl;

cout<<endl;

}

else

{

Pl = Pl->left;

cout<<endl;

cout<<"Wrong!"<<endl;

cout<<endl;

}

Pm = Pm->next;

break;}

case 3:{

cout<<Ph->qt<<endl;

cin>>ansr;

if(ansr == Ph->ans)

{

Pl = Pl->right;

marks = marks + 3;

cout<<endl;

cout<<"Correct!"<<endl;

cout<<endl;

}

else

{

Pl = Pl->left;

cout<<endl;

cout<<"Wrong!"<<endl;

cout<<endl;

}

Ph = Ph->next;

break;

}

}

}

cout<<"Your score: "<<marks<<endl; //final score is printed 5 means fail, 12 means highest grade

return 0;

}