

Solution:

int balanceSum(int \*A, int length) {

int left\_sum=A[0];

int right\_sum=0,i;

for(i=1;i<length;i++)

{

right\_sum=right\_sum+A[i];

}

for(i=1;i<length;i++)

{

right\_sum=right\_sum-A[i];

if(left\_sum==right\_sum)

{

return i+1;

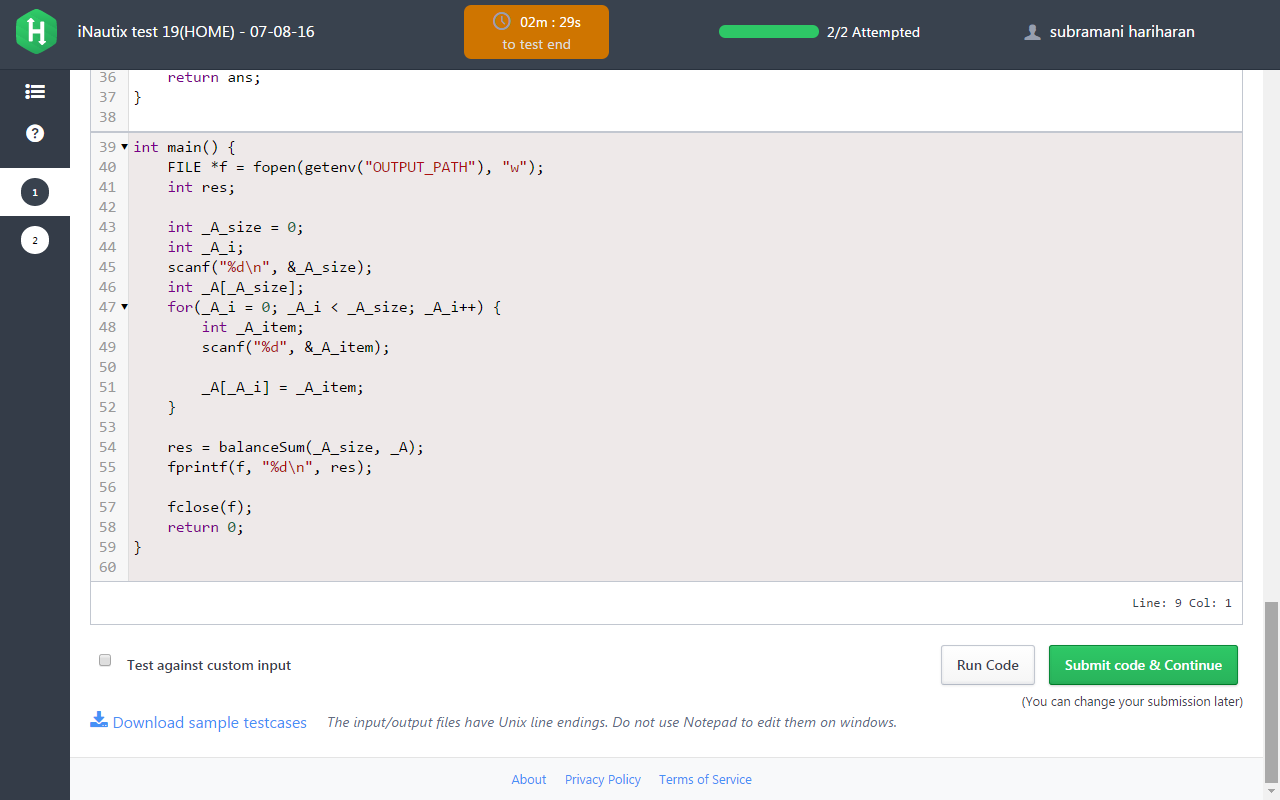
}

left\_sum=left\_sum+A[i];

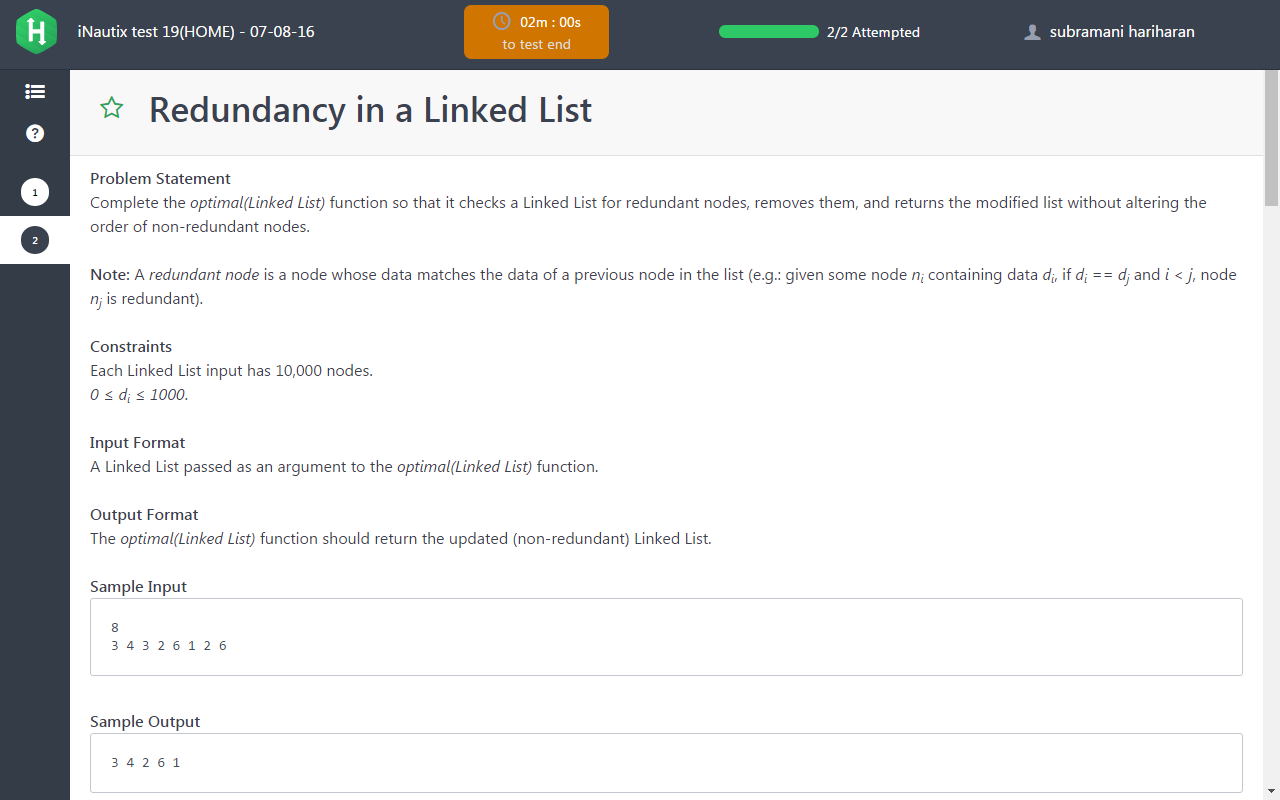
}

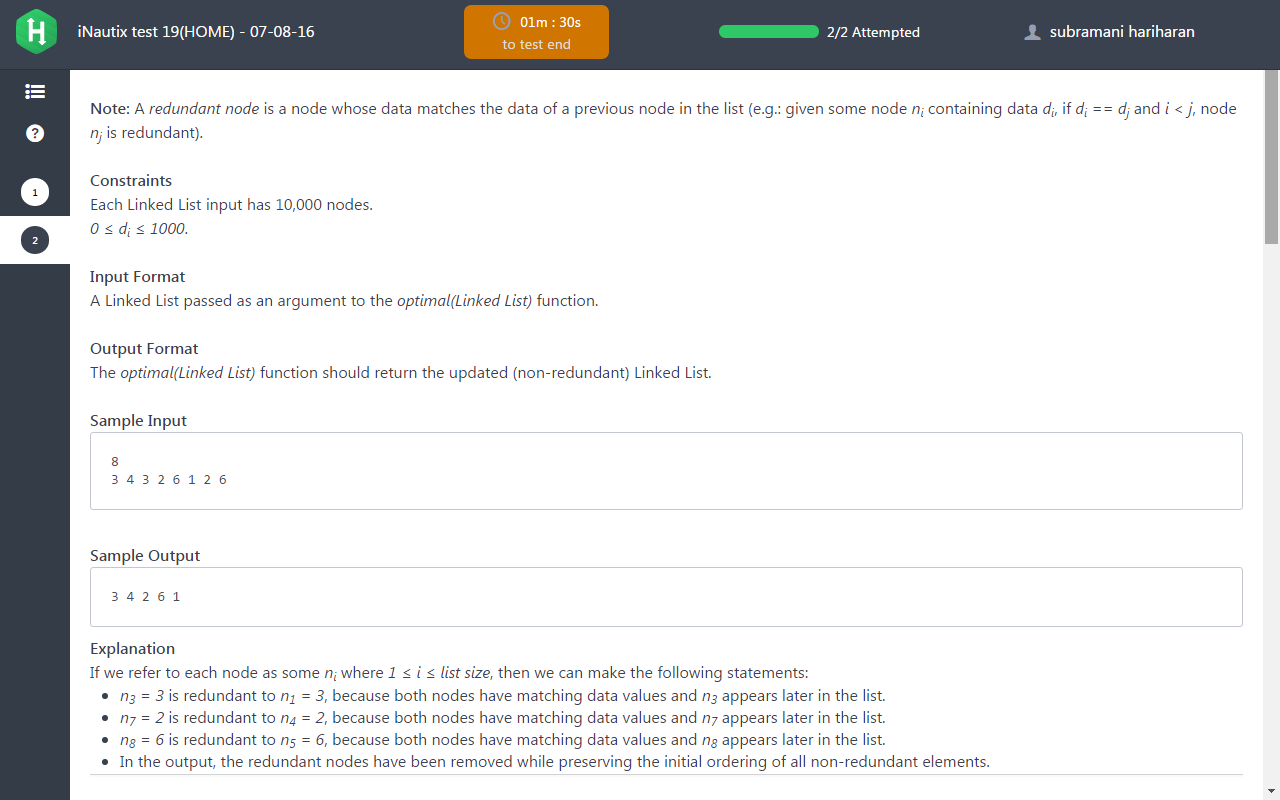
return 0;

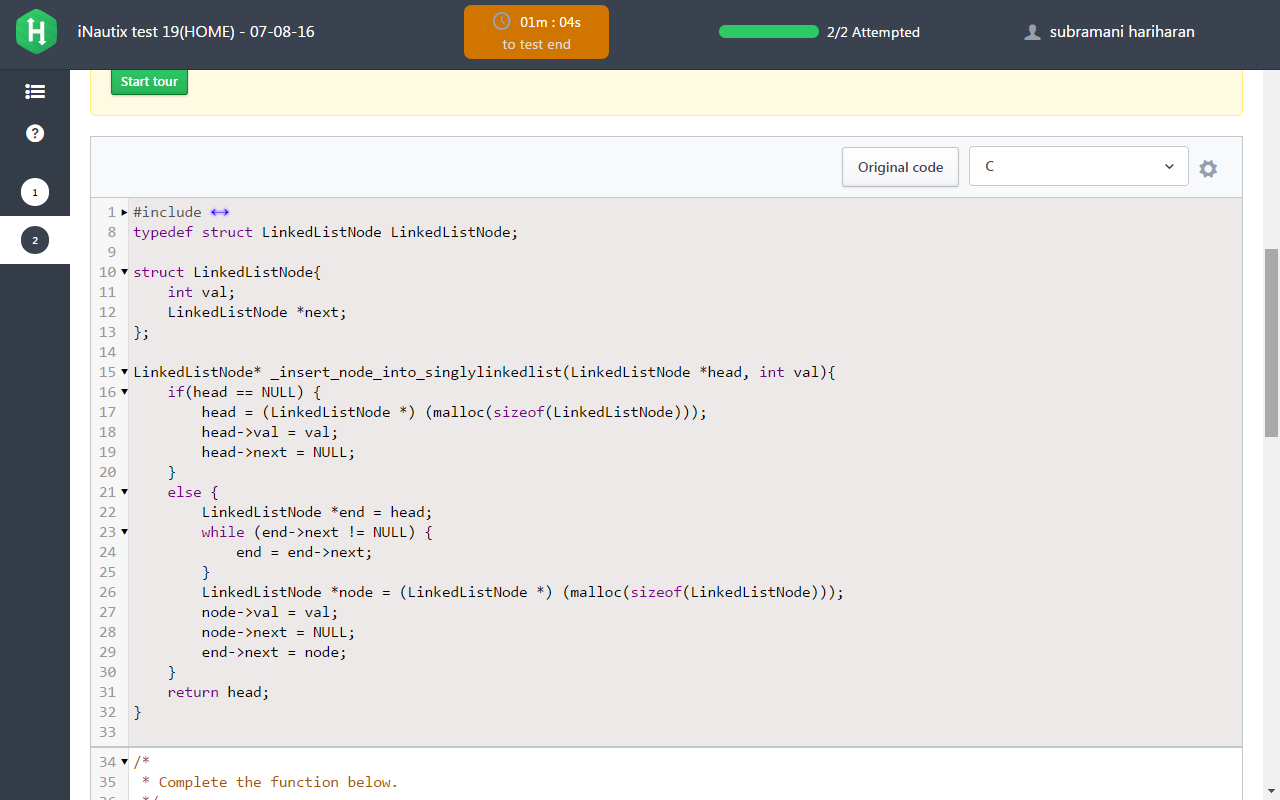
}

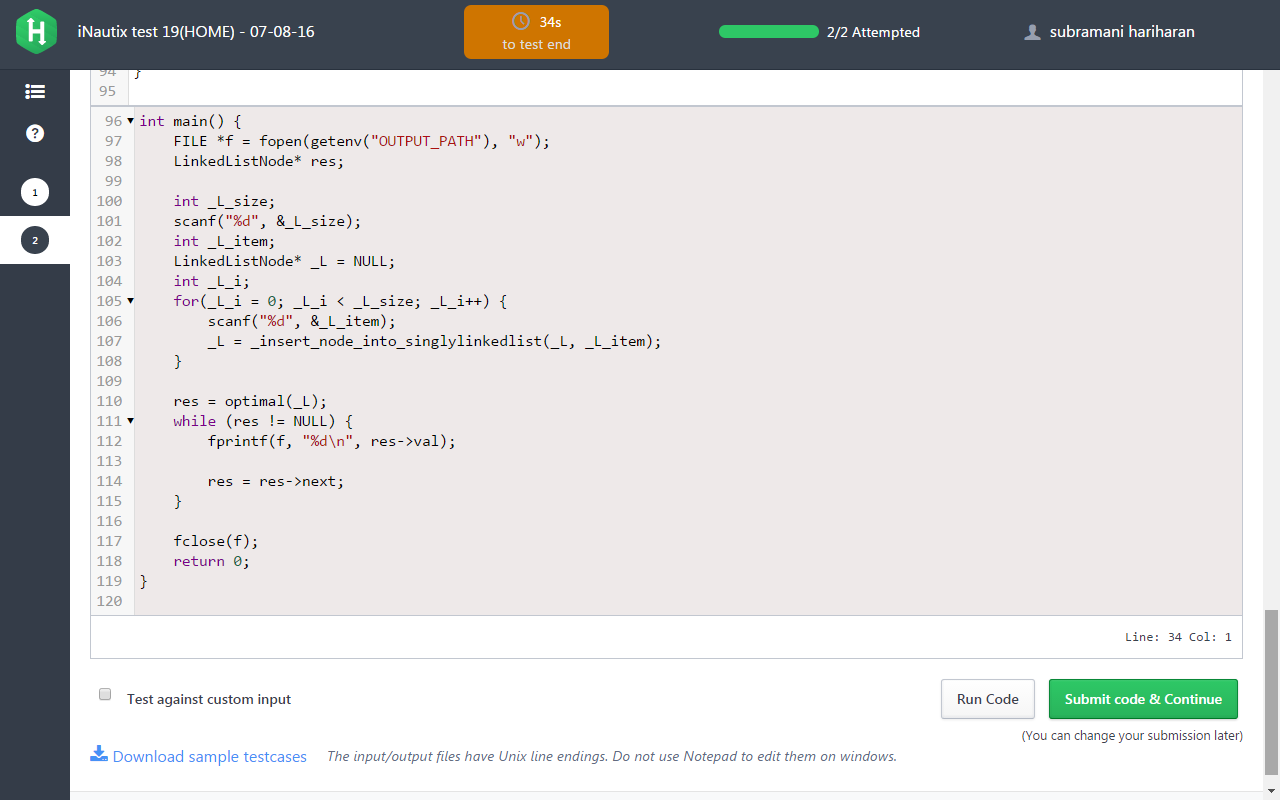


PROBLEM 2:









Solution:

/\*

\* Complete the function below.

\*/

/\*

For your reference:

LinkedListNode {

int val;

LinkedListNode \*next;

};

\*/

LinkedListNode\* optimal(LinkedListNode\* L) {

LinkedListNode \*head=L;

int flagship=0;

int nodes[1000]={0};

while(head!=NULL)

{

if(flagship==0)

{

int temp=head->val;

nodes[temp]=1;

flagship=1;

}

else

{

if(head->next!=NULL)

{

int temp=head->next->val;

if(nodes[temp]==1)

{

head->next=head->next->next;

}

else

{

nodes[temp]=1;

head=head->next;

}

}

else

{

return L;

}

}

}

return L;

}