1. Correct the following code:

Please write in “correct” in the box next to the lines that are correct. If the line is wrong, please correct the entire line. Write the correct line in the provided box next to the line.

Template <class T>

Void ChunkQueue <T>::Enqueue(int value)\_ \_ \_ \_ \_

{

If (head==tail)\_ \_ \_ \_

{

head=new Node;

tail=head;\_ \_ \_ \_ \_--->tail=nullptr

next=nullptr;\_ \_ \_ \_ \_

prev=nullptr;\_ \_ \_ \_ \_

len=0; \_ \_ \_ \_ \_

{

tail--->values[0]=value;\_ \_ \_ \_ \_

len++;//\_ \_ \_ \_

If (tail-->len=<ARRAY\_SIZE)//\_ \_ \_ \_ \_

{

node\*newTail=new Node;

newTail-->next=tail;\_ \_ \_ \_ \_ \_

newTail-->prev=nullptr;\_ \_ \_ \_

newTail-->len=0;

tail-->next=newTail;\_ \_ \_ \_

tail=nullptr;\_ \_ \_ \_

}

}

1. Fill in the blanks

The following program outputs four lines of output.

Write what the output would be for each line:

Line1:”\_ \_ \_ \_ \_ \_”

Line2:”\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_”

Line3:”\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_”

For the purpose of this problem assume that the variables i main are stored at the following memory addresses:

The variable a in the main is stored at address 100

The variable b in the main is stored at address 200

The variable c in the main is stored at address 300

The variable d in the main is stored at address 400

The variable e in the main is stored at address 500

Memory allocated on the heap using new will be stored at address 800

Void mystery (int \*a, int & b) {

B- -;

\*a=6;

caut<<\*a<<””<<b<<endl;

a=&b;

}

Int main () {

Int a=2;

Int b=60;

Int c=200;

Int \*d=new int;

\*d=b;

d=&b;

Int \*e=&c;

Mystery (d,c);

Caut <<a<<””<<b<<””<<c<<””<<d<<””<<e<<endl;

caut<<8a<<””<<8b<<””<<8c<<””<<\*d<<””<<8e<<endl;

Return 0;

}

1. Code

Complete code

Template <class T>

Void Trec<T>:: Add (T elem) {

Node\* curr=\_ \_ \_;

//create newNode

Node\*newNode=\_ \_ \_;

\_ \_ \_=elem;

\_ \_ \_=1;

NewNode-->\_ \_ \_=nullptr;

newnode-->right=\_ \_ \_;

if(\_ \_ \_( )) {

root=\_ \_ \_;

\_ \_ \_++;

return;

}

while (\_ \_ \_):=nullptr) {

If (elem=curr-->value) {

delete NewNode;

\_ \_ \_ ++;

break;

}

else if (elem<curr-->value) {

If (\_ \_ \_) {

curr-->left=\_ \_ \_

break;

} else {

curr=\_ \_ \_;

}

}

else {

If (curr-->right==\_ \_ \_)

\_ \_ \_=newNode;

\_ \_ \_;//found the left

} else (

\_ \_ \_=curr-->right;

}  
}

}

\_ \_ \_++;

}

Template <class T>

Void tree <T>::place In Order (Node \* node) {

If (node==\_ \_ \_) {

return;

}

Place in order (\_ \_ \_);

for (int i=0; i<\_ \_ \_; i ++) {

iterQue.push(\_ \_ \_);

}

Place In Order (\_ \_ \_ \_);