UID: 55696-70018

1. Program does not compile I receive these errors

```
bash-4.2$ ls
pa1.c  sample.c  sample.h
bash-4.2$ gcc -std=c99 -pedantic -Wvla -Wall -Wshadow -g *.c -o pa1
\In file included from pa1.c:1:0:
sample.h:18:1: warning: useless storage class specifier in empty declaration [enabled by default]
 };
 ^
pa1.c: In function ‘main’:
pa1.c:7:11: error: ‘verifying’ undeclared (first use in this function)
           verifying e condition //pseudo, there was failure while getting the options
           ^
pa1.c:7:11: note: each undeclared identifier is reported only once for each function it appears in
pa1.c:7:21: error: expected ‘)’ before ‘e’
           verifying e condition //pseudo, there was failure while getting the options
                     ^
pa1.c:11:5: error: expected expression before ‘FILE’
     FILE *fpr = fopen("operations_input_file.txt", "r");
     ^
pa1.c:16:9: error: ‘fpr’ undeclared (first use in this function)
     if (fpr == NULL)
         ^
pa1.c:20:21: error: ‘EXIT’ undeclared (first use in this function)
         return exit(EXIT EXIT_FAILURE)
                     ^
In file included from sample.h:8:0,
                 from pa1.c:1:
pa1.c:20:26: error: expected ‘)’ before numeric constant
         return exit(EXIT EXIT_FAILURE)
                          ^
pa1.c:21:5: error: expected ‘;’ before ‘}’ token
     } else
     ^
pa1.c:23:19: error: ‘val’ undeclared (first use in this function)
     while (fread(&val, sizeof(int), 1, fpr) == 1 ))
                   ^
pa1.c:23:51: error: expected statement before ‘)’ token
     while (fread(&val, sizeof(int), 1, fpr) == 1 ))
                                                   ^
pa1.c:25:8: error: ‘i’ undeclared (first use in this function)
     ar[i]=fread(&val, sizeof(int), 1, fpr);
       ^
pa1.c:30:11: error: ‘I’ undeclared (first use in this function)
     while(I=1) {
           ^
pa1.c:33:9: error: expected expression before ‘struct’
         struct *node, ar[i])
         ^
```

```
pa1.c:33:9: error: too few arguments to function ‘insert’
In file included from pa1.c:1:0:
sample.h:26:14: note: declared here
 struct node* insert(struct node *root, int x);
              ^
pa1.c:34:9: error: expected ‘;’ before ‘if’
         if (ch[i] == 'd')
         ^
pa1.c:15:10: warning: variable ‘ch’ set but not used [-Wunused-but-set-variable]
     char ch[20];
          ^
pa1.c:14:9: warning: variable ‘ar’ set but not used [-Wunused-but-set-variable]
     int ar[20];
         ^
pa1.c: At top level:
pa1.c:44:1: error: expected identifier or ‘(’ before ‘}’ token
}
 ^
In file included from sample.c:4:0:
sample.h:18:1: warning: useless storage class specifier in empty declaration [enabled by default]
 };
 ^
sample.c: In function ‘new_node’:
sample.c:15:5: warning: statement with no effect [-Wunused-value]
     p->height;
     ^
sample.c: In function ‘Postorder’:
sample.c:223:15: error: ‘null’ undeclared (first use in this function)
     if (root==null) //to terminate the infinite recursion
               ^
sample.c:223:15: note: each undeclared identifier is reported only once for each function it appears in
sample.c: At top level:
sample.c:236:22: error: expected ‘{’ before ‘*’ token
 int evaluate (struct *node, FILE * file_ptr)
                      ^
sample.c: In function ‘deleteNode’:
sample.c:218:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
sample.c: In function ‘evaluate’:
sample.c:239:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
```

2.

3.

4.

5. This user has not left any comments for peer review or marked anything specifically for peer review

6. This code does look as if it is setup to create the tree that it is checking, it looks to use recursion during rotates or in inserts to make sure that the new value goes into the correct location. It does use stacks as well for this. In the placing of a new key in each node it checks it against the parent to make sure that it is on the proper side.

7. This user has created a Balance function and makes sure that he also updates the height given the needed information. This is how they check for balance throughout and this is done in the insert function.

8. It does not appear that the user has a condition setup to return 0 for the first digit if something goes wrong, however the user has made sure to include returning a -1 if the file can not be opened.

9. The user has two arrays set to 20 entries for both integers and characters for the input, while this is good for what we were given, I would caution against and recommend using dynamically allocated arrays if they wanted to go this way.

10. The user created an if statement, if it was greater it would go to the right but if it was less than or equal to it will go to the left child. Once it finds a location it does dynamically allocate memory for the new key. There is a function to delete nodes that also frees up memory when running through everything needed for the BST. The new node is attached as either the right or left child of said parent node, called root, if needed the value in parent is stored in a temp when moving around, it is able to do this with recursion, since you would not want to move it until you are working with a certain parent child subset.

11. The user has setup a set of cases with if statements for the BST to run through to see which set of rotations it does or does not need to go through. It sends this through the Balance function so it looks to find the youngest ancestor on the way down and given recursion it should also store the parent node. For balancing the user uses an algorithm based on the height to compare and see if something is out of place.

12. As in step 11 the Balancing function finds the tallest child to be used, and it does correctly identify when to use each type of rotation. It looks like it does move the middle subtree from child to ya and because the user uses a temp holder variable the outer subtrees remain untouched.

13. As of the code that has been provided the user has not implemented the delete function into the code but it does look like the delete function runs correctly. It uses the nodes key to compare and given what the outcome is it moves the keys appropriately. The delete function only removes one child at a time but if it does have to move two then it saves one in a temp holder variable and is able to use that to not destroy the integrity of the tree. When everything is placed back the overall shape of the tree is still the original as needed.

14. After a deletion the program updates the height which will lead to finding the youngest unbalanced ancestor.

15. The code after deletion is a copy of what is right after the insert.

16. This file does not compile so there is not an output file.