CIS 2101 [Data Structures and Algorithms]

Date Due: Sept 7, 2020 before 7:30 am.

Assignment #2

Topic: Linked List andExecutionStach

Given the definition: **typedef struct node {** **char data;** **struct node *link;** **}nodetype, *List;** **/* Definition of a List */**	**Function Specifications:** 1)Function changeOldLetter(). The function will change the first occurrence of a given old letter to a given new letter in the given list. 2)Function deleteAllOccur(). The function will delete all occurrences of the given letter in the given list. **Hint**: Take note of the word “given” in the problem, it implies that the data is to be passed as a parameter. Determine data is to be passed by address or by copy.

For each of the function above, do the following:

Function changeOldLetter()
1)Write the function Header.

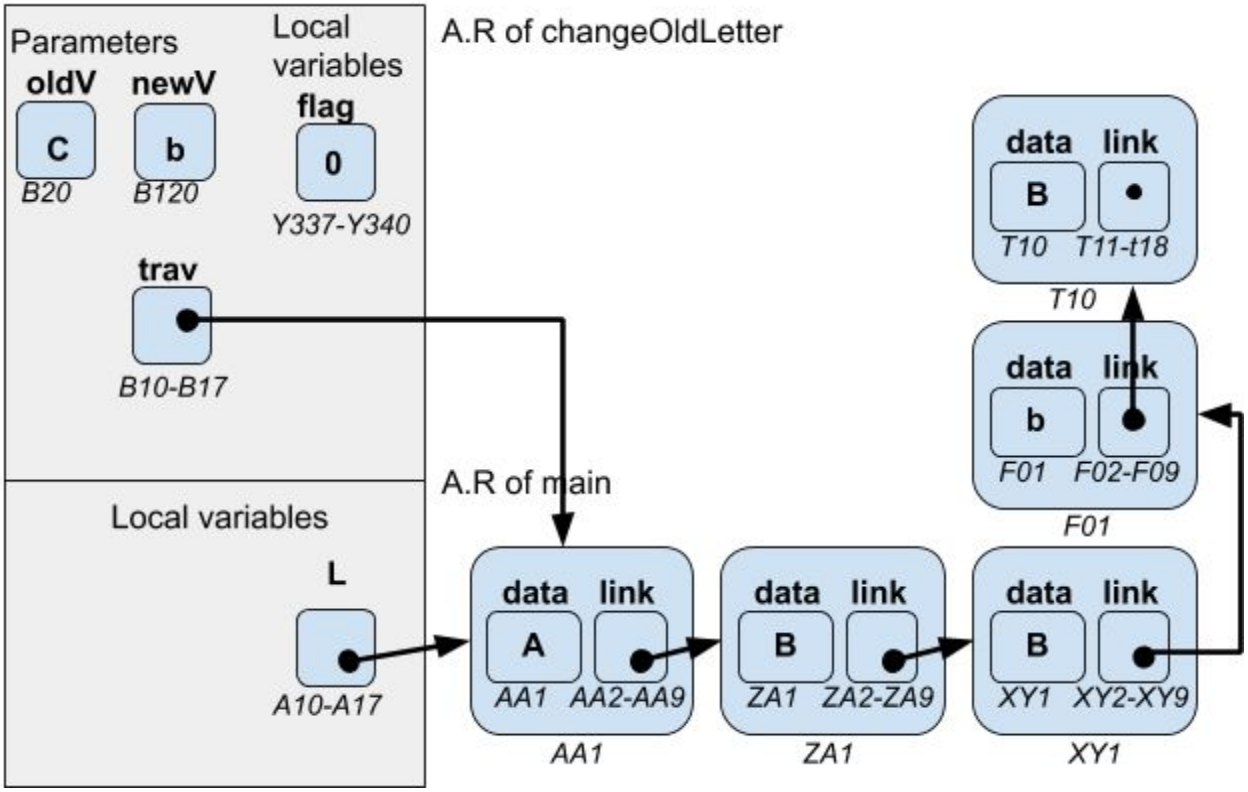
void changeOldLetter(List A, char old, char new);

2)Write a sample function call. Declare and initialize (if necessary), all the variables found in the call BEFORE function call. In the declaration of the List variable, make a comment that it is populated with the letters 'A', 'B', 'B', 'C', 'B'.

a. For function changeOldLetter(): old letter is 'C'
List L; // List L is populated with the letters 'A', 'B', 'B', 'C', 'B'
changeOldLetter(L, 'C', 'b');

3)Assume that the function call is in main, draw the execution stack, illustrating the activation records when the function is called.

a. Draw a rectangle for each of the activation records with activation of main() at the bottom of the exe. stack.
b. Each variable, **draw a box** and label it with:
i. Variable name (no data type)
ii. Beginning address (Arbitrary value, i.e. you may choose an appropriate value)
iii. Value

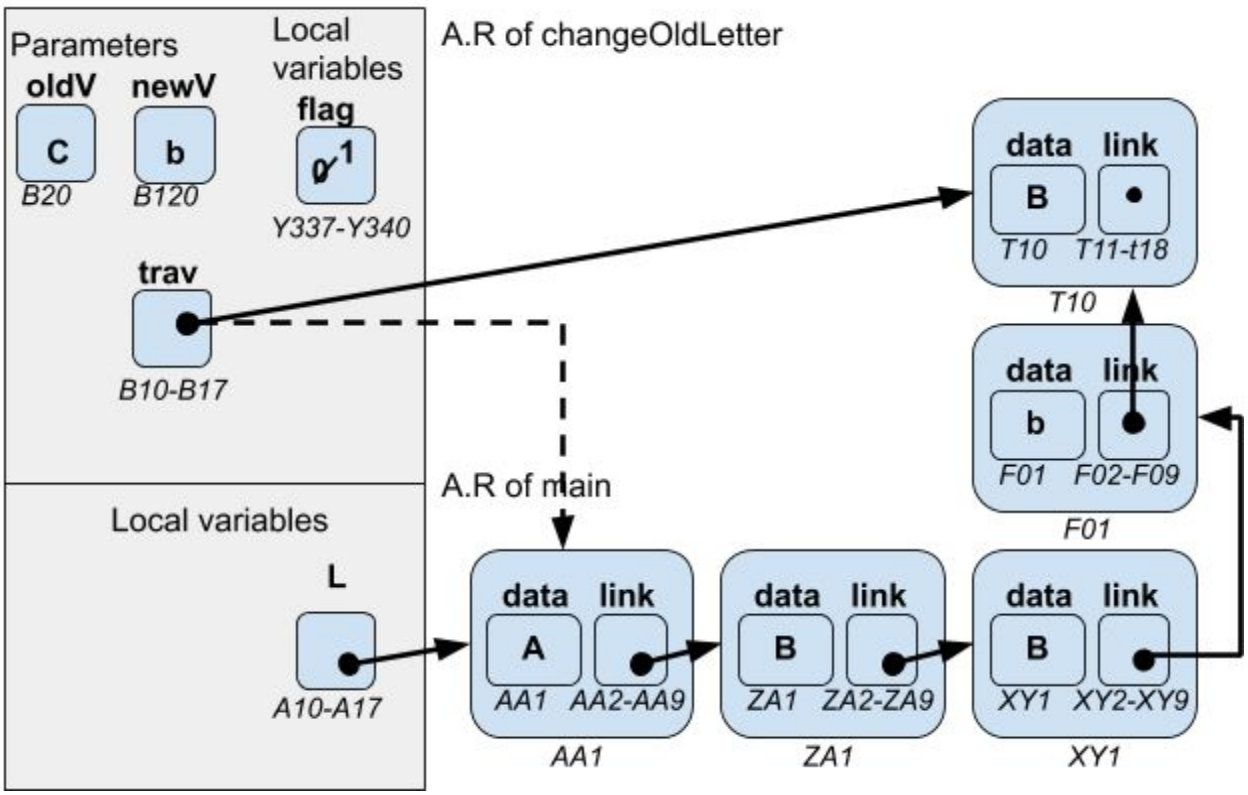


4)Write the function definition (header + body)

```
void changeOldLetter (List A, char old, char new) {
        List trav;
        int flag = 0;

        for (trav = A; trav != NULL && flag == 0; trav = trav->link) {
                if (trav->data == oldV) {
                        trav->data = newV;
                        flag = 1;
                }
        }
}
```

5)Simulate the function using the execution stack in #3, show where the pointer is currently pointing and put a symbol to denote the old value of the pointer.



Function deleteAllOccur()

1)Write the function Header.

void deleteAllOccur (List *A, char val) ;

2)Write a sample function call. Declare and initialize (if necessary), all the variables found in the call BEFORE function call. In the declaration of the List variable, make a comment that it is populated with the letters 'A', 'B', 'B', 'C', 'B'.

b. For function deleteAllOccur(): given letter is 'B'

List L; // List L is populated with the letters 'A', 'B', 'B', 'C', 'B'
Void deleteAllOccur(&L, 'B');

3)Assume that the function call is in main, draw the execution stack, illustrating the activation records when the function is called.

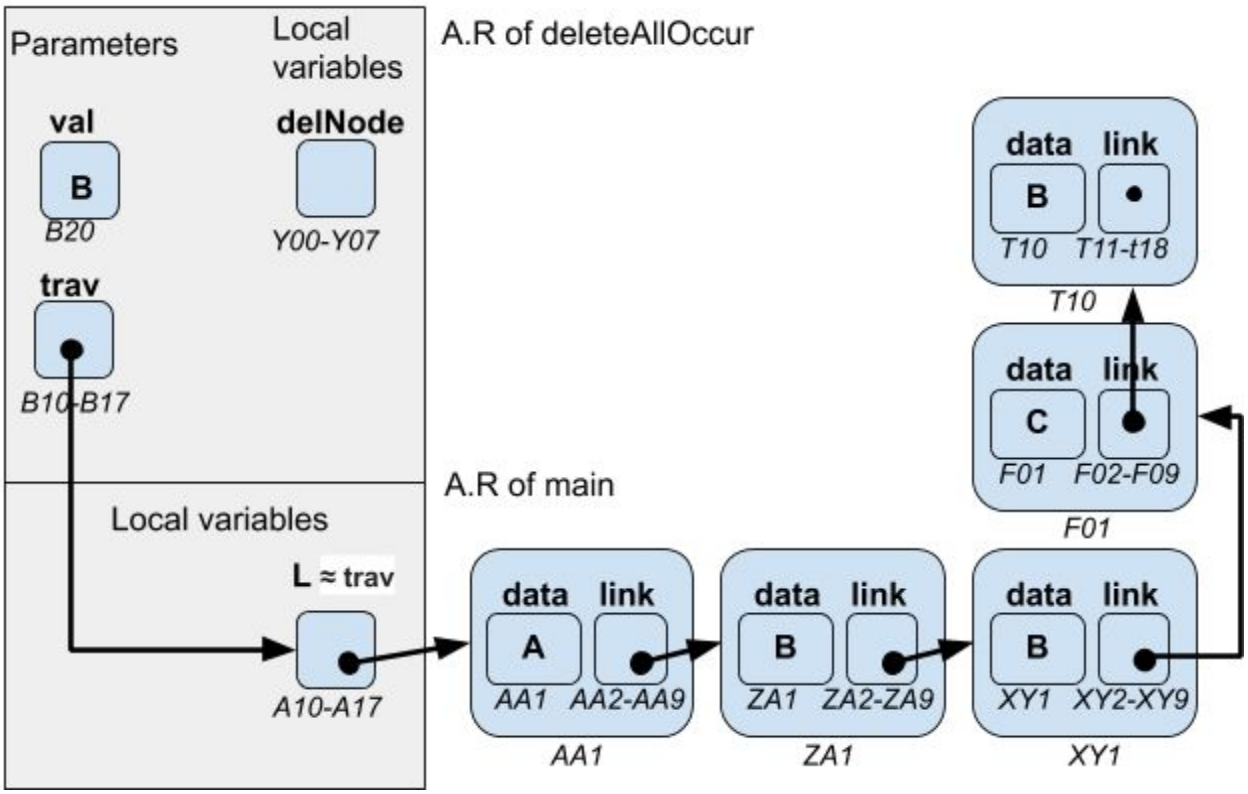
a. Draw a rectangle for each of the activation records with activation of main() at the bottom of the exe. stack.

b. Each variable, **draw a box** and label it with:

i. Variable name (no data type)

ii. Beginning address (Arbitrary value, i.e. you may choose an appropriate value)

iii. Value



4)Write the function definition (header + body)

```
void deleteAllOccur (List *A, char val) {
        List *trav, delNode;
        trav = A;

        while (*trav != NULL) {
                if ((*trav)->data == val) {
                        delNode = *trav;
                        *trav = delNode->link;
                } else {
                        trav = &(*trav)->link;
                }
        }
}
```

5)Simulate the function using the execution stack in #3, show where the pointer is currently pointing and put a symbol to denote the old value of the pointer.