

You may use one page of hand written notes (both sides) and a dictionary. No i-phones, calculators nor any other type of non-organic computer.

Otherwise, returns a value less than zero.

1) Linked List (version 1): This C program compiles and runs. It correctly adds char arrays (names) to a linked list in lexical order. What is its output?

int strcmp (const char * str1, const char * str2)

This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.

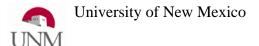
Returns: zero if both strings are equal. A value greater than zero if the first character that does not match has a greater value in str1 than in str2.

```
1) #include <stdio.h>
2) #include <stdlib.h>
3) #include <string.h>
4)
5) #define MAX_NAME_LEN 16
6)
7) struct Node
8) {
9)
   struct Node *next;
10)
   char name[MAX_NAME_LEN];
11) };
12)
13) struct Node *start = NULL;
15) void addNode(char* name);
16)
18) void main(void)
20) {
21)
    addNode("something");
22)
    addNode("wicked");
23) addNode("this");
24)
   addNode("way");
    addNode("come");
25)
26) }
27)
```



```
29) void addNode(char* name)
31) {
32)
     struct Node *newNode = malloc(sizeof(struct Node));
33)
     strcpy(newNode->name, name); //string copy
34)
     if(start == NULL)
35)
     { printf("addNode(%s) NULL", name);
36)
37)
       start = newNode;
38)
       start->next = NULL;
     }
39)
40)
     else if (strcmp(start->name, name) > 0)
41)
42)
     { printf("addNode(%s) START",name);
       newNode->next = start;
43)
       start = newNode;
44)
     }
45)
46)
     else
47)
48)
       printf("addNode(%s) ", name);
49)
       struct Node *node = start;
50)
       int done = 0;
       while(!done)
51)
52)
53)
         printf("%s ", node->name);
         if (node->next == NULL)
54)
55)
         { node->next = newNode;
56)
          newNode->next = NULL;
57)
           done = 1;
         }
58)
59)
         else if (strcmp(node->next->name, name) > 0)
60)
61)
         { newNode->next = node->next;
62)
          node->next = newNode;
           done = 1;
63)
64)
65)
66)
         node = node->next;
67)
68)
69)
     printf("\n");
70) }
71)
```





2) Linked List (version 2): The removeNode function below is added to linked list version 1 and the main function in version 1 is replaced with the main shown below. This version 2 of linked list compiles and runs. However, it segfaults.

Which is the first line (in execution order) on which a segfault might occur? Fix the removeNode function so that it correctly removes nodes.

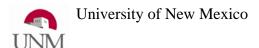
```
2) void removeNode(char* name)
4) {
5)
    struct Node* deletedNode = NULL;
6)
7)
    if (start == NULL) return;
    if (strcmp(start->name, name) == 0)
8)
9)
     deletedNode = start;
10)
     free(deletedNode);
11)
12)
13)
14)
    else
15)
     struct Node *node;
16)
     node = start;
17)
     while(node->next)
18)
19)
20)
       if (strcmp(node->next->name, name) == 0)
21)
22)
        deletedNode = node->next;
        free(deletedNode);
23)
24)
25)
       node = node->next;
26)
27)
   }
28)
29)
30)
31)
32)
34) void main(void)
36) {
37)
    addNode("something");
```

CS 241: Data Organization using C Final Exam



```
38)
      addNode("wicked");
39)
      addNode("this");
      addNode("way");
40)
41)
      addNode("come");
      removeNode("way");
42)
      removeNode("macbeth");
43)
      removeNode("something");
44)
      while(start) removeNode(start->name);
45)
46) }
```

University of New Mexico



3) Bit Operators: This C program compiles and runs. What is its output?

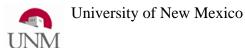
```
1) #include <stdio.h>
 2) void main(void)
 3) { unsigned char x = 103;
 4)
 5)
     unsigned char a = x << 3;
 6)
     unsigned char b = x >> 3;
 7)
     unsigned char c = x & 10;
     unsigned char d = x \& 59;
8)
9) unsigned char e = x | 10;
10) unsigned char f = x ^ 10;;
11)
12) printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n",
13)
             a, b, c, d, e, f);
14) }
```

4) This C program compiles and runs. If the output from lines 7 and 8 is:

```
sizeof(long)=8
x=0x7fff29af6530, x[0]=22
```

Then what is the output from line 10?

```
1) #include <stdio.h>
2)
3) void main(void)
4) {
     long a[] = \{22, 33, 44, 55, 66, 77\};
5)
6)
      long *x = a;
7)
      printf("sizeof(long)=%lu\n", sizeof(long));
      printf("x=%p, x[0]=%d\n", x, x[0]);
8)
      x = x + 3;
9)
10)
      printf("x=%p, x[0]=%d\n", x, x[0]);
11) }
```



5) This C program compiles and runs. What is the output?

```
1)
     #include <stdio.h>
     int level;
 2)
 3)
     void swap(int v[], int i, int j)
 4)
 5)
     \{ int c = v[i];
        v[i] = v[j];
 6)
 7)
       v[j] = c;
     }
 8)
 9)
10)
     void quicksort(int v[], int left, int right)
11)
12)
        level++;
13)
        printf("level=%d: %d, %d\n", level, left, right);
14)
        int i, last;
15)
        if (left < right)</pre>
16)
17)
18)
          swap(v, left, (left+right)/2);
19)
          last = left;
20)
          int num2 = v[left];
21)
          for (i=left+1; i <= right; i++)</pre>
22)
23)
            if (v[i] < v[left])</pre>
            { last++;
24)
25)
              swap(v, last, i);
26)
            }
27)
          }
28)
29)
          swap(v, left, last);
30)
          quicksort(v, left, last-1);
          quicksort(v, last+1, right);
31)
32)
33)
        level--;
34)
35)
36)
37)
     void main(void)
38)
39)
        int v[] = {55, 22, 77, 88, 33, 11};
        int arraySize = sizeof(v)/sizeof(int);
40)
        level = 0;
41)
        quicksort(v, 0, arraySize-1);
42)
43)
```