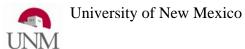


You may use one page of hand written notes (both sides) and a dictionary. No i-phones, calculators nor any other type of non-organic computer.

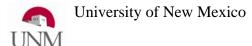
1) Bit Operators: This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
 2) void main(void)
 3) {
 4)
      unsigned char x = 75;
 5)
 6)
      unsigned char a = x << 3;
 7)
      unsigned char b = x >> 3;
 8)
      unsigned char c = x & 12;
 9) unsigned char d = x \& 200;
      unsigned char e = x | 7;
10)
      unsigned char f = x ^ 7;
11)
12)
13) printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n",
14)
              a, b, c, d, e, f);
15) }
```



2) Squeeze: removing a character from a string in place. This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
 2)
 3) void main(void)
 4) {
 5)
      char s[]="AzBBzzCCCzD";
      char del ='z';
 6)
      int srcIdx=0, snkIdx=0;
 7)
 8)
      while (s[srcIdx])
      { if (s[srcIdx] != del)
 9)
        { s[snkIdx] = s[srcIdx];
10)
          snkIdx++;
11)
        }
12)
13)
        else
14)
        { printf("[%d,%d] %s\n", srcIdx, snkIdx, s);
15)
16)
        srcIdx++;
17)
18) }
```

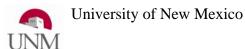


3) Find Substring - using Pointers. This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
2) #include <string.h>
 3)
 4) char *findSubstring(char *str, char *needle)
 5) {
      int len = strlen(needle);
 6)
 7)
      int n = 0;
8)
9)
      while (*str)
     { printf("%c%c\n",*str, *needle); //中午午午午午午午午午
10)
11)
        if ( *(needle+n) == *str)
12)
13)
        { n++;
14)
          if (n == len) return (str-len)+1;
15)
16)
        else
17)
        { str -= n;
18)
          n = 0;
19)
20)
        str++;
21)
      }
22)
      return NULL;
23) }
24)
25) void main(void)
26) {
27)
      char* sub=findSubstring("AforBfooCfoooD","foo");
      printf("==>%s\n",sub);
28)
29) }
```

4) Memory Allocation - malloc. What expression should be placed inside the call to malloc to allocate memory for an int array with n elements?

```
int *num = malloc( ? );
```



5) Binary Tree. This C program compiles and runs. It is supposed to create a binary tree with the letters sorted in depth-first order. However, the resulting tree is out of order. Why? Where does the first miss placement occure?

```
1) #include <stdio.h>
 2) #include <stdlib.h>
 3) #include <string.h>
 4)
 5) struct tnode
 6) { char letter;
 7)
     int count;
 8)
      struct tnode *left;
 9)
      struct tnode *right;
10) };
11)
12)
13)
14) struct tnode *talloc(char c)
15) { struct tnode *node = malloc(sizeof(struct tnode));
16)
17)
      node->letter = c;
18) node->left = NULL;
19) node->right = NULL;
20) node->count = 1;
21)
     return node;
22) }
23)
24)
25)
26)
27)
28) void print(struct tnode *node)
29) {
30)
     if (node == NULL) return;
      print(node->left);
31)
32)
      printf("%c(%d)\n", node->letter, node->count);
      print(node->right);
33)
34) }
```

```
35) struct tnode* addNode(struct tnode *parent, char c)
36) {
37)
      struct tnode* newNode = NULL;
      if (parent == NULL) return talloc(c);
38)
39)
40)
      if (c == parent->letter)
41)
      { parent->count++;
42)
      else if (c < parent->letter)
43)
44)
      { if (parent->left == NULL)
45)
        { parent->left = talloc(c);
46)
        }
47)
        else
48)
        { addNode(parent->left, c);
49)
50)
51)
      else
52)
      { if (parent->right == NULL)
53)
        { parent->right = talloc(c);
54)
55)
        else
        { addNode(parent->left, c);
56)
57)
58)
      }
      return NULL;
59)
60) }
61)
62)
63)
64) void main(void)
65) { struct tnode *root;
66)
67)
      char data[] = "helloworld";
68)
      root = addNode(NULL, data[0]);
69)
      int i;
70)
      for (i=1; i<strlen(data); i++)</pre>
      { addNode(root, data[i]);
71)
72)
73)
      print(root);
74) }
```