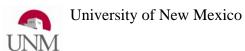


You may use one page of hand written notes (both sides) and a dictionary. No i-phones, calculators nor any other type of non-organic computer.

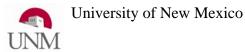
1) If Logic: This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
 2) void main(void)
 3) {
 4)
      int a = 10;
      if (a % 7 == 0)
 5)
 6)
      { printf("A\n");
 7)
        a+=2;
 8)
      }
 9)
      else
10)
      { printf("B\n");
11)
        a+=4;
12)
13)
      printf("C\n");
14)
      if (a % 7 == 0)
15)
      { printf("D\n");
16)
        a+=3;
17)
      }
18)
      else
19)
      { printf("E\n");
        if (a > 5)
20)
21)
        { printf("F\n");
22)
          a +=4;
23)
        }
24)
        else
25)
        { printf("G\n");
26)
          a = 4;
27)
28)
29)
      printf("%d\n",a);
30) }
```



2) Variable Scope: This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
 2)
 3) int a=5;
 4)
 5) int foo(int n)
 6) {
 7)
      int b=2;
 8)
      a++;
 9)
      b++;
10) n = n+a+b;
11)
    printf("foo: a=%d, b=%d, n=%d\n", a, b, n);
12)
      return n;
13) }
14)
15) void main(void)
16) {
17)
     int a, n;
18)
      n = 5;
19)
    a = foo(n);
20) printf("main: n=%d, a=%d\n", n, a);
21)
22) \quad a = foo(n);
23)
      printf("main: n=%d, a=%d\n", n, a);
24) }
```



3a) Binary Search: This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
 2)
 3) int binarySearch(int x, int v[], int length)
 4) {
 5)
      int low, high, mid;
 6)
      low = 0;
      high = length-1;
 7)
 8)
 9)
      while (low <=high)</pre>
10)
        mid = (low+high)/2;
11)
        printf("[%d %d %d] ", low, mid, high);
12)
13)
14)
        if (x < v[mid]) high = mid-1;
        else if (x > v[mid]) low = mid+1;
15)
16)
        else return mid;
17)
18)
      return -1;
19) }
20)
21) void main(void)
22) {
23)
      int nums[] = {12, 13, 15, 17, 21, 23, 27, 39, 43, 51};
24)
      printf("idx = %d\n", binarySearch(39, nums, 10));
      printf("idx = %d\n", binarySearch(10, nums, 10));
25)
26) }
```

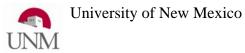
3b) Segmentation Fault: In the code given in part a, if line 24 is changed to:

printf("idx = %d\n", binarySearch(10, nums, 12));

Then binarySearch is called with "bad data". Why is this data "bad"? Even with this bad data, the program will not segmentation fault. Why is it impossible for line 14 or line 15 to give a segmentation fault with this data?

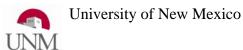
4) Quicksort: This C program compiles and runs. What is its output?

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



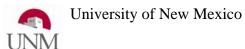
5) Bit Operators: This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
 2) void main(void)
 3) { unsigned char x = 55;
 4)
 5) unsigned char a = x << 4;</p>
 6) unsigned char b = x >> 4;
     unsigned char c = x & 15;
 7)
     unsigned char d = x & 240;
 8)
 9) unsigned char e = x | 15;
10) unsigned char f = x ^ 15;
11)
12) printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n",
             a, b, c, d, e, f);
13)
14) }
```



6) 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[]="Julzzzizzaz";
      char del ='z';
 6)
      int srcIdx=0, snkIdx=0;
 7)
 8)
      while (s[srcIdx])
      { if (s[srcIdx] != del)
 9)
        { s[snkIdx] = s[srcIdx];
10)
11)
          snkIdx++;
        }
12)
13)
        else
14)
        { printf("[%d,%d] %s\n", srcIdx, snkIdx, s);
15)
16)
        srcIdx++;
17)
18) }
```



7) Converting an ASCII char array to an integer: This C program compiles and runs. What is its output?

```
1)
     #include <stdio.h>
 2)
 3)
     int atoiVariant(char s[])
 4)
     {
        int i=0;
 5)
 6)
        int n=0;
 7)
       int sign=1;
 8)
       while(s[i] == ' ' | | s[i] == '\t') i++;
 9)
        if (s[i] == '-') sign = -1;
10)
11)
12)
       while (s[i])
       \{ if (s[i] >= '0' \&\& s[i] <= '9') \}
13)
14)
15)
           int d = s[i] - '0';
            n = 10*n + d;
16)
            printf("d=%d, n=%6d\n", d, n); // <--- printf</pre>
17)
18)
          }
19)
         i++;
20)
21)
       return sign*n;
22)
23)
24)
     void main(void)
25)
       char str[] = " -98-76WW21";
26)
27)
       printf("[%s] = %d\n", str, atoiVariant(str));
      }
28)
```