Clearly print (not sign) your name:

_____

You may use one page of handwritten notes (both sides) and a non-electronic dictionary. **No** phones, calculators, tablets, laptops or other microprocessors are allowed. Write your answers on the exam. You may use extra scratch paper (blank, ruled or graph).

Usually, a box around the answer is not necessary; however, if you think your answer might be difficult to distinguish from any intermediate scratch work you might have written on the page, then draw a box around the part you want graded as your answer.

If your answer is correct, then any scratch work is ignored. However, if your answer is wrong, then any scratch work you provide may be useful for awarding partial credit. If you choose to use extra scratch paper, you may keep it or turn it in with your exam. If you choose to turn in extra scratch paper, then ***print your name*** at the top of each page and label scratch work with the question number to which it pertains.

This exam is designed for a 75 minute period.

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

```
 1) #include <stdio.h>
 2) void main(void)
 3) { unsigned char x = 60;
 4)
 5)   unsigned char a = x << 1;
 6)   unsigned char b = x & 24;
 7)   unsigned char c = x & 124;
 8)   unsigned char d = x | 65;
 9)   unsigned char e = x | 22;
10)   unsigned char f = x ^ 22;
11)
12)   printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n",
13)           a, b, c, d, e, f);
14) }
```

2) Bubble Sort. This C program is intended to sort the values in `main()`'s `numList[]` from highest to lowest. The program compiles and runs; *however, it does not work as intended.* What is the output?

```
 1) #include <stdio.h>
 2)
 3) void swap(int array[], int i, int k)
 4) {
 5)   printf("swap: %d <--> %d\n", array[i],array[k]);
 6)   int tmp = array[i];
 7)   array[i] = array[k];
 8)   array[k] = tmp;
 9) }
10)
11)
12) void bubbleSort(int array[], int n)
13) {
14)   int i, done = 0, start = 0;
15)   while(!done)
16)   {
17)     done = 1;
18)     for (i=start; i<n-1; i++)
19)     {
20)       if (array[i] < array[i+1])
21)       {
22)         swap(array, i, i+1);
23)         done = 0;
24)       }
25)     }
26)     start++;
27)   }
28) }
29)
30)
31) void main(void)
32) {
33)   int numList[] = {7, 3, 2, 9, 5, 3, 8};
34)   int n = sizeof(numList)/sizeof(int);
35)   bubbleSort(numList, n);
36) }
```

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

```
 1) #include <stdio.h>
 2) #include <string.h> //for strlen(s) returns the length of s.
 3) int exponentiation(int base, int exponent)
 4) {
 5)   int i;
 6)   int power = 1;
 7)   for (i=0; i<exponent; i++) power *= base;
 8)   return power;
 9) }
10)
11) void main(void)
12) {
13)   char pentList[] = "0000";
14)   int n=117;
15)   int place = exponentiation(5,strlen(pentList)-1);
16)   int digit;
17)
18)   int i=0;
19)   while(pentList[i])
20)   {
21)     printf("n=%3d, place=%d\n", n, place); //<<<<<<<<<<<<<<<
22)     digit = n / place;
23)     n = n - digit*place;
24)     place = place / 5;
25)     pentList[i] = '0' + digit;
26)     i++;
27)   }
28)   printf("%s\n", pentList);                  //<<<<<<<<<<<<<
29) }
```

4) 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[]="XbXXytXXXe";
 6)   char del ='X';
 7)   int srcIdx=0, snkIdx=0;
 8)   while (s[srcIdx])
 9)   { if (s[srcIdx] != del)
10)     { s[snkIdx] = s[srcIdx];
11)       snkIdx++;
12)     }
13)     else
14)     { printf("[%d,%d] %s\n", srcIdx, snkIdx, s);
15)     }
16)     srcIdx++;
17)   }
18)   s[snkIdx]='\0';
19)   printf("==>%s\n",s);
20) }
```

5) 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;
11)   printf("[%d, %d]\n", left, right);    //<=================
12)   if (left >= right) return;
13)
14)   swap(v, left, (left+right)/2);
15)   last = left;
16)   for (i=left+1; i <= right; i++)
17)   {
18)     if (v[i] < v[left])
19)     { last++;
20)       swap(v, last, i);
21)     }
22)   }
23)
24)   swap(v, left, last);
25)
26)   quicksort(v, last+1, right);
27)   quicksort(v, left, last-1);
28) }
29)
30)
31) void main(void)
32) {
33)   int v[] = {88, 77, 11, 44, 55, 22, 33};
34)
35)   int arraySize = sizeof(v)/sizeof(int);
36)   quicksort(v, 0, arraySize-1);
37) }
```