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?

```
#include <stdio.h>
void main(void)
{
  int a = 10;
  if (a % 7 == 0)
  { printf("A\n");
    a+=2;
  }
  else
  { printf("B\n");
    a+=4;
  }
  printf("C\n");
  if (a % 7 == 0)
  { printf("D\n");
    a+=3;
  }
  else
  { printf("E\n");
    if (a > 5)
    { printf("F\n");
      a +=4;
    }
    else
    { printf("G\n");
      a -=4;
    }
  }
  printf("%d\n",a);
}
```

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

```c
#include <stdio.h>

int a=5;

int foo(int n)
{
  int b=2;
  a++;
  b++;
  n = n+a+b;
  printf("foo: a=%d, b=%d, n=%d\n", a, b, n);
  return n;
}

void main(void)
{
  int a, n;
  n = 5;
  a = foo(n);
  printf("main: n=%d, a=%d\n", n, a);

  a = foo(n);
  printf("main: n=%d, a=%d\n", n, a);
}
```

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;
 7)   high = length-1;
 8)
 9)   while (low <=high)
10)   {
11)     mid = (low+high)/2;
12)     printf("[%d %d %d] ", low, mid, high);
13)
14)     if (x < v[mid]) high = mid-1;
15)     else if (x > v[mid]) low = mid+1;
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));
25)   printf("idx = %d\n", binarySearch(10, nums, 10));
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?

```c
#include <stdio.h>
void swap(int v[], int i, int j)
{
  int c = v[i];
  v[i] = v[j];
  v[j] = c;
}

void quicksort(int v[], int left, int right)
{ int i, last;
  printf("[%d, %d]\n", left, right);
  if (left >= right) return;

  swap(v, left, (left+right)/2);
  last = left;
  for (i=left+1; i <= right; i++)
  {
    if (v[i] < v[left])
    { last++;
      swap(v, last, i);
    }
  }

  swap(v, left, last);
  quicksort(v, left, last-1);
  quicksort(v, last+1, right);
}


void main(void)
{
  int v[] = {55, 22, 77, 88, 33, 11};

  int arraySize = sizeof(v)/sizeof(int);
  quicksort(v, 0, arraySize-1);
}
```

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

```c
#include <stdio.h>
void main(void)
{ unsigned char x = 55;

  unsigned char a = x << 4;
  unsigned char b = x >> 4;
  unsigned char c = x & 15;
  unsigned char d = x & 240;
  unsigned char e = x | 15;
  unsigned char f = x ^ 15;

  printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n",
          a, b, c, d, e, f);
}
```

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

```c
#include <stdio.h>

void main(void)
{
  char s[]="Julzzzizzaz";
  char del ='z';
  int srcIdx=0, snkIdx=0;
  while (s[srcIdx])
  { if (s[srcIdx] != del)
    { s[snkIdx] = s[srcIdx];
      snkIdx++;
    }
    else
    { printf("[%d,%d] %s\n", srcIdx, snkIdx, s);
    }
    srcIdx++;
  }
}
```

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

```c
#include <stdio.h>

int atoiVariant(char s[])
{
  int i=0;
  int n=0;
  int sign=1;

  while(s[i] == ' ' || s[i] == '\t') i++;
  if (s[i] == '-') sign = -1;

  while (s[i])
  { if (s[i] >= '0' && s[i] <= '9')
    {
      int d = s[i] - '0';
      n = 10*n + d;
      printf("d=%d, n=%6d\n", d, n);   // <--- printf
    }
    i++;
  }
  return sign*n;
}

void main(void)
{
  char str[] = " -98-76WW21";
  printf("[%s] = %d\n", str, atoiVariant(str));
}
```