

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

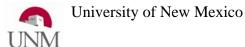
```
1) #include <stdio.h>
 2)
 3) int binarySearch(int x, int v[], int high)
 4) {
 5)
      int mid;
 6)
      int low = 0;
 7)
      while (low <=high)</pre>
 8)
 9)
10)
        mid = (low+high)/2;
        printf("[%d %d] ", low, high);
11)
12)
13)
        if (x < v[mid]) high = mid-1;
14)
        else if (x > v[mid]) low = mid+1;
        else return mid;
15)
      }
16)
17)
      return -1;
18) }
19)
20) void main(void)
21) {
22)
      int nums[]={12, 33, 45, 47, 53, 55, 59, 73, 91, 93};
23)
      int n = sizeof(nums)/sizeof(int);
24)
      printf("idx = %d\n", binarySearch(47, nums, n));
25)
      printf("idx = %d\n", binarySearch(88, nums, n));
26) }
```



University of New Mexico

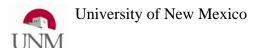
2) Bit Operators: This C program compiles and runs. What is its output?

```
1) #include <stdio.h>
2) void main(void)
3) { unsigned char x = 73;
4)
5) unsigned char a = x << 4;
6) unsigned char b = x >> 4;
7) unsigned char c = x & 7;
8) unsigned char d = x & 99;
9) unsigned char e = x | 7;
10) unsigned char f = x ^ 7;
11)
12) printf("a=%d, b=%d, c=%d, d=%d, e=%d, f=%d\n", a, b, c, d, e, f);
14) }
```



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("Axffoffoofoo","foo");
28)
      printf("==>%s\n",sub);
29) }
```



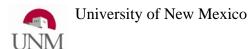
4) This C program compiles and runs. If the output from lines 7 and 8 is: sizeof(long)=8 x=0x7fff29af6530, x[0]=22 Then what is the output from line 10?

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

5) This C program compiles and runs. If executed with the command: a.out 00110023

Then what is the output?

```
1)
     #include <stdio.h>
 2)
 3)
     void main(int argc, char *argv[])
     { if (argc == 2)
 4)
 5)
        \{ int n = 0; 
          char *c_pt = argv[1];
 6)
          while (*c_pt)
 7)
 8)
          { if (*c_pt < '0' || *c_pt > '1') break;
            n = n*2 + *c_pt-'0';
 9)
10)
            c_pt++;
11)
12)
          printf("%d\n", n);
        }
13)
      }
14)
```



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

```
1)
     #include <stdio.h>
 2)
     struct Point {int x; int y;};
 3)
 4)
 5)
     void addToPoint(struct Point *a, struct Point b, int n)
     \{ (*a).x += n; \}
 6)
 7)
       a->y
               += n;
8)
       b.x
               += n;
9)
       b.y
               += n;
     }
10)
11)
12)
     void main(void)
     { struct Point p1 = {7, 7};
13)
       struct Point p2 = {11, 11};
14)
15)
       addToPoint(&p1, p2, 2);
       printf("p1=(%d, %d)\n", p1.x, p1.y);
16)
17)
       printf("p2=(%d, %d)\n", p2.x, p2.y);
     }
18)
```

7) Memory Allocation - malloc. What expression should be placed inside the call to malloc to allocate memory for an double array with n rows of m columns?

```
double *x = malloc( ? );
```