



You may use one 8½×11 inch page of notes (both sides) and a dictionary.

Time: 75 minutes.

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

```
1. #include <stdio.h>
2. int main()
3. { int i=0;
4.   float a = 100;
5.   a = a*a*a*a*a;
6.
7.   float c = 2;
8.   float x = 1000000*c + a;
9.   float y = a;
10.  float z = 0;
11.
12.  for (i=0; i<1000000; i++)
13.  { y += c;
14.    z += c;
15.  }
16.
17.  z += a;
18.  x /= 10000;
19.  y /= 10000;
20.  z /= 10000;
21.
22.  printf("x=%.1f  y=%.1f,  z=%.1f\n", x, y, z);
23. }
```



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

```
1. #include <stdio.h>
2. int main()
3. { unsigned char x=88;
4.
5.     unsigned char a = x << 4;
6.     unsigned char b = x >> 4;
7.     unsigned char c = x & 15;
8.     unsigned char d = x & 240;
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",
13.          a, b, c, d, e, f);
14. }
```



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

```
1. #include <stdio.h>
2.
3. int foo(int n)
4. { int i;
5.   int answer = 0;
6.   for (i=1; i<=n; i++)
7.     { answer += i;
8.       printf("%d, ", answer);
9.     }
10.  return answer;
11. }
12.
13. int main()
14. { printf("%d\n", foo(7));
15. }
```

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

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



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

```
1. #include <stdio.h>
2. #include <string.h>
3.
4. char *getSubstring(char *s1, char *s2)
5. { int len = strlen(s2);
6.   int n = 0;
7.   while (*s1)
8.     { printf("%d:%c%c ",n, *s1, *(s2+n));
9.       if ( *(s2+n) == *s1)
10.        { n++;
11.          if (n == len) return (s1 - len)+1;
12.        }
13.       else
14.        { s1 -= n;
15.          n = 0;
16.        }
17.       s1++;
18.     }
19.   return NULL;
20. }
21.
22. int main()
23. { char a[] = "aab";
24.   char b[] = "axyaaabz";
25.   printf(" ==> %s\n",getSubstring(b,a));
26.   printf(" ==> %s\n",getSubstring(a,b));
27. }
```



```
1. #define MAX_LIST_ITEMS 20
2.
3. struct linkedList
4. { int start; //index of first record
5.   int free; //index of first unused record.
6.   double data[MAX_LIST_ITEMS]; //linked list data
7.   int next[MAX_LIST_ITEMS]; //index of next record in
8.                               //either used or free list.
9.                               //-1 indicates the end in either list.
10. } list;
11.
12. //Inserts value in linked list after beforeMeIdx.
13. //Returns -1 on error.
14. int insertInList(double value, int beforeMeIdx)
15. { if (beforeMeIdx >= MAX_LIST_ITEMS) return -1;
16.
17.   int newIdx = list.free;
18.   list.free = list.next[list.free];
19.   list.data[newIdx] = value;
20.   if (beforeMeIdx < 0)
21.   { //Insert at start of list.
22.     list.next[newIdx] = list.start;
23.     list.start = newIdx;
24.   }
25.   else
26.   {
27.     ???
28.     ???
29.   }
30.   return 0;
31. }
```

6) The code above defines the structure of a linked list and has a function that inserts an item in the list. What code could be placed in lines 24 and 25 in order to complete the list insertion?

7) There is a possible segmentation fault in this code: where and why?