

1. (10 pts) Give the output for the following program:

```
#include <stdio.h>

void f(char *x, char *y) {
    printf("%s\n", x);
    return;
    printf("%s\n", y);
}

int main() {
    char *x = "Well begin, is";
    char *y = "half done!";
    f(x, y);
    return 0;
}
```

2. (10 pts) Give the output for the following program:

```
#include <stdio.h>

void f(int x, int *y) {
    x = 99;
    *y = 99;
}

int main() {
    int m = 17;
    int n = 88;
    f(m, &n);
    printf("(m, n) is: (%d, %d)\n", m, n);
    return 0;
}
```

3. (10 pts) Give the output for the following program:

```
#include <stdio.h>
#define MAX 5

int main() {
    int x = 25;
    int *p = &x;
    *p *= MAX;
    printf("x is: %d\n", x);
    return 0;
}
```

4. (10 pts) In the program below, a print function that prints the array n is called on line 16, but it does not appear in the program. Write the print function.

```
1 #include <stdio.h>
2 #define SIZE 10
3
4 void compute_fibonacci(int x[]) {
5     int count = 2;
6     x[0] = 1;
7     x[1] = 1;
8     for ( count = 2; count < SIZE; ++count ) {
9         x[count] = x[count-1] + x[count-2];
10    }
11 }
12
13 int main() {
14     int n[SIZE];
15     compute_fibonacci(n);
16     print(n);
17     return 0;
18 }
```

5. (10 pts) The following program segment contains code to initialize an array with 10 random numbers. A stub for a function to find the largest element in the array is given on lines 11 to 22. Supply the missing lines of code so that the function **largest** returns the largest element in the array x . Your function **largest** should consist of a single loop.

```
1 #include <stdio.h>
2 #define SIZE 10
3
4 void initialize(int x[]) {
5     int count;
6     for ( count = 0; count < SIZE; ++count ) {
7         x[count] = rand() % 100;
8     }
9 }
10
11 int largest(int x[]) {
12
13
14
15
16
17
18
19
20
21
22 }
23
24 int main() {
25     int n[SIZE];
26     initialize(n);
27     printf("The largest is %d\n", largest(n));
28     return 0;
29 }
```

6. (10 pts) A code segment is listed below with a `swap` function and a `sort` function. Write the code for the `sort` function so that it implements either bubble sort or selection sort.

```
void swap(int *x, int *y) {
    int t = *x;
    *x = *y;
    *y = t;
}

void sort(int a[]) {
}
```

7. (10 pts) The program segment below builds a two-dimensional array `tree` of characters that, when printed, will look like a triangle. Write code for function `print_tree` so that it prints the contents of array `tree` so that it looks like a triangle.

```
#include <stdio.h>
#define ROWS 7
#define TREE_ROWS 5
#define COLUMNS 2*TREE_ROWS

void init(char tree[ROWS][COLUMNS]) {
    int i, j;
    for (i = 0; i < ROWS; ++i) {
        for (j = 0; j < COLUMNS; ++j) {
            tree[i][j] = ' ';
        }
    }
}

void make_tree(char tree[ROWS][COLUMNS]) {
    int i, j;
    int spaces = TREE_ROWS - 1;
    int stars = 1;
    for (i = 0; i < TREE_ROWS; ++i) {
        for (j = 0; j < spaces; ++j) {
            tree[i][j] = ' ';
        }
        for (j = spaces; j < spaces+stars; ++j) {
            tree[i][j] = '*';
        }
        --spaces;
        stars += 2;
    }
}

int main() {
    char tree[ROWS][COLUMNS];
    init(tree);
    make_tree(tree);
    print_tree(tree);
    return 0;
}
```

8. (10 pts) Assuming your `print_tree` function works, what is the output of the above program.