

Fall 2023 COP 3502 Section 4 Final Exam - Part A (25 pts) 12/7/2023

Last Name: _____, **First Name:** _____

UCFID: _____, **Lab Number:** _____

For this portion of the exam, you'll complete a solution to the following problem:

There are n points of interest on the Cartesian grid for this problem. You are an Uber driver starting at the first point listed amongst these n points of interest. You will make $n-1$ trips, visiting each of the other locations exactly once. The amount of money you make for a single trip is the square of the distance between the two points you are traveling. Since you want to maximize profit, the location you visit from your current location will always be the farthest location that has yet to be visited, from your current location. If there is a tie between multiple locations, go to the one with the minimum x coordinate. If there is still a tie, go to the one with the smallest y coordinate. Here is the input/output format:

First line will contain a single positive integer, n ($2 \leq n \leq 1000$), the number of points of interest.

The following n lines will contain two space separate integers each: x_i ($0 \leq x_i \leq 10^4$) and y_i ($0 \leq y_i \leq 10^4$), respectively, representing the x and y coordinate of the i^{th} point ($1 \leq i \leq n$). The output will be a single integer, the money you'll make following the directions stated.

Here is a skeleton of the solution, including function prototypes and some parts of main:

```
#include <stdio.h>
#include <stdlib.h>

long long distsq(long long* pt1, long long* pt2);
int farthest(int curI, int* used, long long** pts, int n);
int beat(long long** pts, int curI, int i, int j);

int main() {

    int n;
    scanf("%d", &n);
    long long** pts = calloc(n, sizeof(long long*));
    for (int i=0; i<n; i++) pts[i] = calloc(2, sizeof (long long));

    for (int i=0; i<n; i++)
        scanf("%lld%lld", &pts[i][0], &pts[i][1]);

    /* Allocate and initialize used array. (Question 2) */
    int curI = 0;
    long long res = 0;

    for (int i=0; i<n-1; i++) {
        /* Fill in code for traveling each subsequent path. (Question 4) */
    }

    printf("%lld\n", res);
    /* Free dynamically allocated memory. (Question 5) */
    return 0;
}
```


Fall 2023 COP 3502 Section 4 Final Exam - Part B (100 pts) 12/7/2023

Last Name: _____, **First Name:** _____

UCFID: _____, **Lab Number:** _____

1) (10 pts) Write a function that takes in a positive integer n , dynamically allocates a 2D integer array of size $(2n+1)$ by $(2n+1)$, fills it with a design of integers in between 0 and n , labeling which “level” a cell is from the center, and returns a pointer to the array. For example, for $n = 2$, the array should be filled as follows:

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Fill in the function prototype below. You may use any functions from the math library as needed.

```
int** levelsquare(int n) {
```

```
}
```

2) (10 pts) Write a function that takes in a pointer to a linked list and uses it to create a new linked list that stores the first, third, fifth, etc. values in the input linked list in the original order, returning a pointer to the front of this newly created list. **Write your function recursively.** (For example, if the input list to the function stores 2, 6, 5, 7, 3, 9 then your function will create a new list storing 2, 5, 3, returning a pointer to the node storing 2. If the input list stored 12, 10, 13, then your function will create a new list storing 12, 13, returning a pointer to the node storing 12.) Watch out for NULL pointers!!!

```
typedef struct node {
    int data;
    struct node* next;
} node;

node* oddrankedterms(node* front) {
```

```
}
```

3) (6 pts) Evaluate the value of the following postfix expressions. No need to show the stack, you'll only be graded on the final answer.

(a) 12 3 / 2 5 + 9 3 - + * _____

(b) 1 2 3 4 5 6 + * + * + _____

(c) 2 3 4 * + 6 4 - / 5 * _____

4) (4 pts) What does the function `g(12, 123)` return, where `g` is defined below?

```
int g(int a, int b) {
    if (b == 0) return 0;
    int res = a * (b & 1);
    return res + g(a, b>>1) ;
}
```

5) (8 pts) Consider the problem of determining in how many whole seconds an object dropped from a given height hits the floor. Any such object has a current height, `h`, a current downward velocity, `v`, and a current downward acceleration, `a`, where `a` is assumed to be constant. In one second from this state, the new height will be `h - v` and the new velocity will be `v + a`. This continues, of course, until the height calculation becomes 0 or negative, indicating that the object hit the floor at that instant. Write a **recursive function**, `hitfloor`, which will take in the parameters described above, `h`, `v` and `a`, as ints (assume `a` is positive), and return the number of whole seconds until the object hits the floor. (Note: Your base case should return 0 for any input indicating that an object is already on the ground or below it.)

```
int hitfloor(int h, int v, int a) {
```

```
}
```

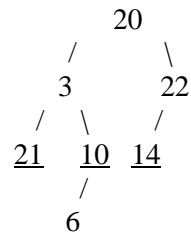
6) (10 pts) Please give **an exact solution** (a function that $T(n)$ satisfies in terms of n) to the following recurrence relation using the iteration technique:

$$T(n) = T(n - 1) + n2^n, \text{ for } n > 1$$
$$T(1) = 0$$

7) (6 pts) Show the contents of the following array after each merge in Merge Sort.

Original	17	13	7	19	22	2	15	14
After 1 st								
After 2 nd								
After 3 rd								
After 4 th								
After 5 th								
After 6 th								
Last	2	7	13	14	15	17	19	22

8) (10 pts) Consider the following problem: given a pointer to the root of a binary tree, and a non-negative integer, k, find the sum of all the values in the nodes that are exactly k levels below the root node. For example, in the tree below, the sum of the values 2 levels below the root is $21 + 10 + 14 = 45$.



Write a function that takes in a pointer to the root of a binary tree and the value of k and returns this sum. Your function **must be recursive**.

```

typedef struct treenode {
    int data;
    struct treenode* left;
    struct treenode* right;
} treenode;

int sumdepthk(treenode* root, int k) {

}
  
```

9) (8 pts) Show the result of inserting each of the following items into an initially empty AVL tree in this order: 9, 12, 20, 2, 5 and 4. **Draw a box around the state of the tree after EACH insertion completes. So you should have six boxes around 6 trees.**

10) (10 pts) Consider the task of printing out the (in lowercase letters) the **first word** alphabetically stored in a trie. (You may assume if a link exists in a trie, that some word exists further down that link.) Write a **void recursive** function to accomplish this task. (Note: Different recursive calls will print different letters in this word.)

```
typedef struct trienode {
    int isWord;
    struct trienode next[26];
} trienode;

void printfirst(trienode* root) {
```

```
}
```

11) (5 pts) Convert 782 in base 10 to base 4.
