Name: TIEN DUNG LE 1002024297 Section: 002

CSE 3318 - Homework 1

Total: 100 points Topics: time complexity of loops, C review: pointers, dynamic memory allocation (with malloc() or calloc()), 1D arrays, pass-by-reference, Valgrind and debugging demo.

What to submit

Submit 3 items: a video file, the updated a C program (scores.c), and a pdf with written answers.
Submit demo video under Homework1-Video in Canvas.
Submit TC.pdf and scores.c in Homework1-Code-TC in Canvas

Video (11 pts)

Provided file to use in your video: demo_sum.c
Record and submit in Canvas a short video that shows you using a Unix system (omega, VM, Ubuntu) with Valgrind and debugging the provided program demo_sum.c .

a) (5pts) On omega or VM or Ubuntu
    a. compile with -g
    b. run with Valgrind
b) (6pts) show the debugging steps listed below. Use either an IDE (VSCode/Code::Blocks/onlinegdb/etc) or the command line debugger from Unix. Set a break point, start running the program in debug mode:
    a. examine value of some variables.
    b. Step line-by-line a couple of times,
    c. set a break point later on (e.g. outside of a loop) and use "continue" to run until it meets that break point.

Check that the video is clear, not blurry. We have to be able to read the text on the screen (commands, value of variables being examined, lines of code, etc).
See a sample video in Canvas. You can use the Studio tool in Canvas (in left menu) to record a screen capture video.
I used Code::Blocks for debugging. It does not have to be Code::Blocks for you. Another IDE (even online) or gdb in command line are also good. The important thing is to show that you can start debugging.

Program(15 pts)

Purpose: 1D array practice, dynamic memory allocation (i.e. using malloc()/calloc()), test functions.

Provided program: scores.c

In scores.c, fill in the code for function:

```
int* get_scores_below(int thresh, int sz_arr, int * arr, int* sz_res)
```

It should return a dynamically allocated 1D array containing values from arr that are strictly less than the given thresh value. This array must be allocated on the heap (malloc/calloc), and its size, sz_res, is updated using pass-by-reference.

If the new array is empty (there are no elements that are less than the threshold) the function returns NULL and sets sz_res to 0.

Examples:

- For threshold 60 and array {92, 57, 100, 95, 38, 10, 85, 91, 20} it should return the array {57, 38, 10, 20}
- For threshold 50 and array {22, 45, 30, 49, 38} it should return the array {22, 45, 30, 49, 38}
- For threshold 22 and array {22, 45, 30, 49, 38} it should return NULL (and not allocate any space). Here there is no value strictly smaller than the threshold.

The function comments in the scores.c file describe the function behavior. Read them.

Hints:

- File scores.c runs a few hardcoded tests for this function. The elements in the result array should appear in the same order they were in the original array. If not, the tests for correctness will fail.
- It is ok to iterate over the input array 2 times:
    - iterate over it once to count how many values are smaller than the threshold, so that you know the size of the result array.
    - allocate the space for the result array
    - iterate again to copy the needed values into the result array
- Pay attention to the indexes in the input and result array. Trace the data on paper! If needed use a debugger to see the values of the indexes (or print them) and compare those with what you expect on paper.

Grading criteria subtask 1 (15 pts)

3 pts – allocate dynamic memory for the exact number of elements smaller than threshold
3 pts – if result array is empty sets sz_res to 0 and returns NULL
3 pts – correctly copy elements in order in which they are in the array
3 pts – sz_res is correctly updated (when the result array is not empty)
3 pts – no Valgring errors

TC questions (74 pts)

Write all your answers for this part in a pdf or docx called TC.pdf and submit it in Canvas.
Fill in NAME and section at the top. Make your answers visible: use a different color or highlight them.
You can handwrite the answer on paper and scan the paper as pdf, but make sure your handwriting is neat and can be read. If the scan is not legible, or did not capture all of your answers on a page, or it is missing a page, you will NOT have a chance to resubmit after the deadline. DOWNLOAD it back from CANVAS and CHECK that the SUBMITTED PDF is good.
You can write your answers on white paper, but they must match the required format (e.g. the tables and the components for the time complexity of loops calculation).
You do not need to 'fit' your answer in the given spacing. You can use as much space as your need.

Q1. (**9** points)
a) (3pts) What is wrong with the time complexity in the statement below?
"Function `void helper(int X, double T);` has time complexity O(V)"

The statement is incorrect because it doesn't provide enough information to determine the time complexity of the function. The time complexity of a function depends on how the inputs are processed within the function and the size of the inputs. The time complexity notation O(V) refers to a function whose time to completion grows linearly with the size of the input V. However, the function void helper (int X, double T); has two inputs, X and T, and without knowing how they are used within the function, it's not possible to determine the time complexity accurately.

Find the dominant term(s) and write O for each of the math functions below.

b) (3pts) $NS + N^3 + 500N^2 + SN^2 + 10^6$ = O(N^3)

c) (3 pts) $100S^3+20S^2+15\lg N +5S$ = O(S^3)

Q2. (65 points) Show your work as done in class. Clearly write the summation and its closed form.

(See cheat sheet for summations. E.g. $1+2+3+\ldots+(n-1)+n = \sum_{t=1}^{n} t$, has closed form: $\frac{n(n+1)}{2}$.)

a) (5 points)

Assume that `void mistery(int X);` has time complexity O(X)

Fill in the time complexity of the function call: `mistery(8);`

$TC_{\texttt{mistery(8)}}$ = O(8)

b) (10 points)

```
for(k = 1; k <= N; k=6*k)
   for(j = 0; j < S; j++)
     printf("D");
```

Iter (e)	j	$TC_{1iter}(j)$ =
0	0	Θ(1)
1	1	Θ(1)
2	2	Θ(1)
3	3	Θ(1)
…		
e	j = e	Θ(1)
…		
p		

change of variable: j = p

j_last = s-1 p = s-1

Sum : $\sum_{j=0}^{S-1} O(1)$ =O(S)

TC of entire loop is O(S)

Iter (e)	k	$TC_{1iter}(k)$ =
0	1	Θ(S)
1	6	Θ(S)
2	36	Θ(S)
3	216	Θ(S)
…		
e	k = 6^e	Θ(S)
…		
p	P	Θ(S)

change of variable: k = p

k_last = N p = log6(N)

Sum: $\sum_{k=1}^{\lfloor \log_6(N) \rfloor} O(S)$

TC of entire loop is $O(\log_6(N))$

Final answer: $O(S \cdot \log_6(N))$

c) (10 points)

```
for(j = N; j >= 0; j-- )
   for(u = 0; u <= j; u = u+9)
      printf("C");
```

Iter (e)	u	$TC_{1iter}(u)$ =
0	0	O(1)
1	9	O(1)
2	18	O(1)
3	27	O(1)
…	…	O(1)
e	u = 9e	O(1)
…	…	…
p	9p	O(1)

change of variable: u = 9k

u_last = j p = (j/9)+1

Sum: $\sum_{u=0}^{j} O(1) = O(j)$

TC of entire loop is O(j)

Iter (e)	j	$TC_{1iter}(j)$ =
0	N	O(N)
1	N-1	O(N-1)
2	N-2	O(N-2)
3	N-3	O(N-3)
…		
e	j = 0	O(1)
…		
p	p	O(1)

change of variable: j = p

j_last = 0 p = N+1

Sum

$$\sum_{j=N}^{0} O(N) = O\left(\frac{2N(N+1)}{2}\right) = O(N(N+1)).$$

TC of entire loop is O(N(N+1))

Final answer: O (N^2)

d) (10 points)

```
if ( check(T)) {   // O(T)
     doThis(M, T)   // O(T^2).
}
else {
     doThat(T)   // O(1)
}
```

Final answer: worst O(T^2) best O(1) in general O(T^2)

Show your work as done in class.

In the worst case, where check(T) is true and we execute doThis(M, T), the overall time complexity is O(T + T^2), and we take the highest term, so it becomes O(T^2).

In the best case, where check(T) is false and we execute doThat(T), the overall time complexity is O(1).

In general, considering both cases, the worst-case time complexity dominates, so the overall time complexity is O(T^2).

e) (10 points)

Assume that `void some_fct(int X);` has time complexity Θ(X) . Solve

```
for(v = 0; v <= N; v = v+1)
   some_fct(v);
```

Fill in: $TC_{\texttt{some\_fct(v)}} = \Theta(X)$

Iter (e)	V	$TC_{1iter}(v)$ =
0	0	Θ(0)
1	1	Θ(1)
2	2	Θ(2)
3	3	Θ(3)
…	…	Θ(…)
e	v = e	Θ(e)
…		
p	V=p	Θ(p)

change of variable: v = v+1

v_last = N p = N-1

Sum = $\sum_{v=0}^{N} \Theta(v)$ = Θ(X) + Θ(X) + … + Θ(X) (N+1)

TC of entire loop is O(X * (N + 1))

Final answer O(X * (N + 1))

f) (10 points)

Assume that `void some_fct2(int N, int k);` has time complexity Θ(N^2)

Solve:

```
for(k = 1; k <= M; k++) {
   some_fct2(k,N);
}
```

Fill in:

$TC_{\texttt{some\_fct2(k,N)}}$ = O(N^2)

Iter (e)	k	TC1iter(k) =
0	1	Θ(N^2)
1	2	Θ(N^2)
2	3	Θ(N^2)
3	4	Θ(N^2)
…		Θ(N^2)
e	k = M	Θ(N^2)
…		Θ(N^2)
p	P+1	Θ(N^2)

change of variable: j = k
j_last = M p = M

Sum: $\sum_{k=1}^{M} \Theta(N^2)$

TC of entire loop is O(M*N^2)

Final answer: O(M*N^2)

g) (10 points) Assume that `void some_fct3(int N);` has TC(N) = Θ(N^3) Find TC for:

```
for(k = 1; k <= M; k=k+1) {
   some_fct3(S);
   for(t = 0; t < M; t = t+1)
      printf("X");
}
```

Fill in: $TC_{\texttt{some\_fct3(S)}}$ = Θ(S^3)

Iter (e)	t	TC1iter(k) =
0	0	Θ(1)
1	1	Θ(1)
2	2	Θ(1)
3	3	Θ(1)
…	…	Θ(1)
e	t = e	Θ(1)
…		Θ(1)
p	p	Θ(1)

change of variable: t = p
t_last = M-1 p = M
Sum $\sum_{k=1}^{M} O(M) = M \cdot O(M)$
TC of entire loop is O(M^2)

Iter (e)	k	TC1iter(k) =
0	1	Θ(M)
1	2	Θ(M)
2	3	Θ(M)
3	4	Θ(M)
…		Θ(M)
e	k = m	Θ(M)
…		Θ(M)
p	P+1	Θ(M)

change of variable: k = p
k_last = M p = M
Sum:__: $\sum_{k=1}^{M} O(M) = M \cdot O(M)$

TC of entire loop is O(M^2 *S^3)

Final answer: O(M^4)