CPSC 223 Spring 2025
Practice Midterm #1

April 24, 2025

Instructions: Write your name and NetID on your answer sheet. You may use one 8½×11” crib sheet (both sides). No electronic devices. Show all work for partial credit. For multiple-choice questions, fill in the box.

Problem 1: Basics (15 points)

(a) (3 pts) Given the files below, mark an X next to each file that contains the machine code for the function `process_data` after running `make`.

```
dataio.h  dataio.c  dataio.o  main.c
main.o  process_data.c  utils.c  utils.h
```

(b) (5 pts) Complete the code below: fill blanks (1)–(4) with letters (A)–(F).

```
typedef struct {
  char *name;
  int score;
} Player;

Player *p = NULL;
/* 1 */
p->name = strdup("Alice");
/* 2 */
printf("%s:␣%d\n", p->name, p->score);
free(/* 3 */);
/* 4 */

A. p = malloc(sizeof *p); B. p = calloc(1, sizeof *p);
C. p = malloc(sizeof(Player)); D. p->score = 0;
E. free(p->name); F. free(p);
```

(c) (4 pts) Write the prototype for a function `apply` that takes an array of `double` of length `n` and a function pointer `double→double`, and returns a newly allocated `double*` with transformed values.

(d) (3 pts) Explain what happens if you call `free` twice on the same pointer. Why is this dangerous?

(e) (3 pts) Given this typedef:

```
typedef void (*op_fn)(int*, int);
```

Write a function `map_array` that takes an `int* arr`, its length `n`, and an `op_fn` to apply to each element in place.

Problem 2: Algorithms (20 points)

(a) (6 pts) For each snippet, give the tightest Big-O in terms of n:

- (a) `for(int i=0;i<n;i++) for(int j=i+1;j<n;j++) work();`
- (b) `int i=1; while(i<n) work(); i*=2;`
- (c) `qsort(a,n,sizeof *a,cmp);`
- (d) `linear_search(a,n,key);`
- (e) `merge_sorted(x,m,y,m);`

(b) (6 pts) Describe in-place partitioning in quicksort, and why worst-case is $\Theta(n^2)$.

(c) (8 pts) Sorted array sum-to-T in $\Theta(n)$: give two-pointer pseudocode and explain correctness.

Problem 3: Arrays, Array Lists, Linked Lists (20 points)

(a) (8 pts) Table: for each operation, fill in Θ-time for (i) fixed C array, (ii) doubling array list, (iii) singly-linked list with head only.

Operation	Array	Array List	Singly-Linked
Append at end			
Remove at index i			
Get element at index i			
Insert at front			

(b) (6 pts) Draw the memory diagram after:

```
int *a = malloc(4*sizeof *a);
for(int i=0;i<4;i++) a[i]=i+1;
int *b = a+2;
int c[4];
for(int i=0;i<4;i++) c[i] = a[i]*2;
```

(c) (6 pts) Write `push_front` and `pop_front` for a singly-linked list of `int` with global `node* head`.

Problem 4: Trees (15 points)

(a) (5 pts) Draw BST inserting M, C, T, A, J, P, X in that order.

(b) (5 pts) What insertion order yields minimum-height? What yields maximum-height?

(c) (5 pts) Complete this C function for tree height:

```
int tree_height(node *r) {
  if(r==NULL) return 0;
  int hl = /*1*/;
  int hr = /*2*/;
  return /*3*/;
}
```

Fill in (1)–(3).

Problem 5: Stacks, Queues, Hashtables (15 points)

(a) (6 pts) For each ADT, state Θ-time of core ops: stack (push/pop), queue (enqueue/dequeue), hashtable (insert/lookup).

(b) (4 pts) Given ring-buffer queue:

capacity=8, head=5, tail=2
indices: 0 1 2 3 4 5 6 7
array: $[, , X, , , A, B, C,]$

Show state after: enqueue D; dequeue twice; enqueue E, F.

(c) (5 pts) Implement `bool contains_cycle(node* head);` using tortoise and hare.

Problem 6: Practice Exam Review (10 points)

(a) (5 pts) For snippets below, write `INVALID`, `UNDEFINED`, or the output. Assume variables set up as in class examples.

(a) `b++; printf("$% c`
`n", b[0]);`

(b) `(*b)++; printf("$% \n", a);`

(c) `c = &b[0]; printf("$% s`
`n", *c);`

(b) (5 pts) For each data structure below, give the asymptotic time of `remove_incoming(v)` in a directed graph with V vertices and E edges: (i) adjacency matrix, (ii) adjacency list.

Problem 7: ADT Implementation (10 points)

Complete the C function to remove every other element from a doubly-linked list (head/tail pointers), no leaks:

```c
void list_thin(list *l) {
  list_node *n = l->head;
  while(n && n->next) {
    list_node *n2 = n->next->next;
    /* A: free second node */
    /* B: relink pointers to skip freed node */
    if(l->tail == n->next) l->tail = n;
    n = n2;
  }
}
```

Fill in A and B with proper statements.

Problem 8: Graphs (15 points)

(a) (7 pts) True/False: Incidence graph of the Fano plane is nonplanar because it contains a subdivision of $K_{3,3}$. Justify or sketch.

(b) (8 pts) Write Dijkstra's algorithm in pseudocode, showing initialization, relax, and priority queue ops.

Good luck!