Final Exam
ESE 124 – Programming Fundamentals
Fall 2023
Instructor: Jenny Chen

ID: First Name: Last Name:

On the final exam, you have to answer **at least 3 out of 4 questions**. If you choose to answer **all 4, you can earn 20 points(additional extra credits).** Just circle the 3 questions you want to answer. If you **don't circle any, we'll assume you're picking the first 3 questions. Brief comment is required on your code (10% of your grade).**

---

1) [**33 points**] Solving the problem:

Develop a C program that accomplishes text encryption and decryption by shifting each letter to the next in sequence, and others remain. For example, 'A' would be transformed into 'B,' 'J' into 'H,' 'z' into 'a' and so forth. The program should be designed to read an input file, perform text encryption, decrypt the text back to its original form, and employ **dynamic memory allocation** for both processes.

Programming Implementation (23 points):

```
// Function prototypes
char* encrypt(char* input);
char* decrypt(char* input);
```

File Handling (5 points):

Documentation and Comments (5 points):

Input:

xyz aaB NIb 12Yafh

12cbsz @ yza 123azy

output:

Encrypted result:

yza bbC OJc 12Zbgi

12dcta @ zab 123baz

Decrypted result:

xyz aaB NIb 12Yafh

12cbsz @ yza 123azy

2) [**33 points**] Solving the problem:

Given a string A, containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid, meaning that the brackets must close in the correct order, e.g., "()" and "()[]{}" are all valid but "(]" and "([)]" are not. You must use the ADT stack to solve the problem. The string is read from an input file.

input1:

{ 09 [(})]

output:

unbalanced

input2:

{ 09 [()]}

output:

balanced

3) [**33 points**] Solving this problem required **dynamic memory allocation, struct car and struct boat**:

Loading cars onto two types of ferry boats. One boat works like a stack, and the other like a queue. Cars preference is in the original **cars.txt file**. When both boats arrive, the order in which cars leave is based on when they boarded. The Stack boat lets the last 'S' car off, and the Queue boat starts with the first 'Q' car. For the program: 1. Input boat capacities.. 2. Create structures for Car and Boat. 3. Loading the cars onto the boats. 4. Unloading the cars onto the boats. You might need functions like print stack/queue, manage boat, and so on. **The order in which cars leave from the Boat S are stored in "stack.txt". The order in which cars leave from Boat Q are stored in "queue.txt"**.

Cars.txt(license plates and boat preference "%s %c"):

ZZA9896 S

MIN5689 Q

NHI9098 S

KNM4545 Q

IPO8987 Q

Output:

Enter the capacity of boat S: 4

Enter the capacity of boat Q: 3

stack.txt

Car ID: NHI9098, Boat #: S

Car ID: ZZA9896, Boat #: S

queue.txt

Car ID: MIN5689, Boat #: Q

Car ID: KNM4545, Boat #: Q

Car ID: IPO8987, Boat #: Q

4) [**33 points**] Solving the problem:

In a vending machine, one product costs 15 cents. It accepts nickels and dimes. Coins other than these trigger 'b = 1' and are returned. If total < 15 cents, it waits. If a coin exceeds 15 cents, the product delivery and extra coin is returned. When total = 15 cents, 'p = 1' for product delivery. Pressing 'C' cancels and returns entered coins. UI sets 'n = 1' for nickels, 'd = 1' for dimes, and 'c = 1' for cancel. The FSM sets 'b = 1' for non-nickel/dime coins, 'r = 1' for coin return, and 'p = 1' for product delivery. The dispense mechanism follows FSM commands for 'b,' 'r,' and 'p.

You need to provide a **diagram: 5 points, next states and output table: 5 points, and C code with comments: 23 points.**