Time taken (before correction and edits): 1 : 13

Question 1

1. Learning how to use different software and then combine them all together to make a software system, where they all work towards a common cause.
   To create an application distributed across multiple languages & runtime environments.

2. The client is who sends out the packet to the server for what document/program they want to use. The client side is weaker, runs the requested program. All users are clients.
   The programs running on the user's browser

3. The back end is all the code that supports the web server that is not seen visually, for example, scripts, CGI and anything parsing input from the webpage. It is the opposite of the front end which consists of the visual things like HTML.
   The programs running on the server.

4. A packet consists of a sender address (who sent the packet), a receiving address (where is the packet being sent) and an instruction.
   Data structure with from/to address & data. Used to communicate a message over a network.

5. A web server consists of a server that is deployed to the web, making its contents viewable. Hosts webpages and their content?
   Program that waits for request from client, executes request and returns answer to client.

6. A session consists of when one person logs in (to a Unix OS), including when they're executing commands and goes all the way up to when they logout.
   Period of time at server between request arrival & returned reply.

7. An interpreter takes source code and interpretes in a way such that it can give direct instructions to the CPU, without having to translate all of the program first.
   Program that executes a script without compilation.
   An example would be Python. Bash, Python

8. A compiler converts source code into machine language (binary), so that the CPU can understand and process the instructions.
   Converts a source file into a binary file, directly executable by machine.
   The GNU C Compiler, GCC is an example. C, Java

Question 2: CGI

```
<html>
  <header> <title> CGI </title></header>
  <body>
    <!-- Large Title-->
    <h1> Order Your Food Here </h1>
    <!-- Form that calls ProcessOrder.py -->
    <form action="ProcessOrder.py" method="POST">
      <input type="checkbox" name="Hotdog" value="1">
      Hotdog
      <br>
      <input type="checkbox" name="Hamburger" value="1">
      Hamburger
      <br>
      <input type="checkbox" name="Side Order" value="1">
      Side Order
      <input type="radio" name="Side" value="1">
      Fries
      <input type="radio" name="Side" value="2">
      Onion Rings
      <br>
      <input type="checkbox" name="Drink" value="1">
      Drink
      <select name="Dr">
      <option value="1"> Coke </option>
      <option value="2"> Diet Pepsi</option>
      <option value="3">7UP</option>
      <option value="4">Orange</option>
      <option value="5">Water</option>
      </select>
      <br>
      <input type="submit" value="SUBMIT">
    </form>
  </body>
</html>
```

Question 3: GNU Makefile The makefile targets don't work as makefiles don't have the same conditional notation as BASH.

```
main: main.o library.o menu.o order.o report.o
# All .o files must exist
# Compile under main
	gcc -o main main.o main.o library.o menu.o order.o report.o

main.o: main.c main.h library.h menu.h order.h report.h
	gcc -c main.c

order.o: order.c order.h
	gcc -c order.c

report.o: report.c report.h
	gcc -c report.c

library.o: library.c library.h
	gcc -c library.c

menu.o: menu.c menu.h
	gcc -c menu.c

backup:
	if [!-d backup] then #If backup doesn't exist
		mkdir backup # Create the new dir
	fi
# copy all .c and .h files into backup
	cp *.c *.h backup

tar:
	if [!-d backup] then # Backup doesn't exist
		echo"Backup directory does not exist,
        nothing was TAR'ed"
	else
	cd backup # Go to backup dir
	tar -cf * backup.tar # Tar all files
```

```
rm *.c *.h # Remove all .c and .h files
```

Question 4: C Programming

```
int parse (char line[], char buf1[], char buf2[],
           char buf3[], int bufSize){
  // Line is a comma separated String
  // 3 buffers are empty

  char c = line[0]; // Dummy char for condition
  int fields = 1; // How many fields we have
  // Start at 1 since commas indicate fields+1

  int i= 1; // 1 since we already got first char
  int j = 0;
  int k,l = 0; // 4 indices for 4 Strings
  while(c!='\0'){// Loop until end of String
    if(c==',')fields++; // Comma, inc fields
    else if(fields==1 && j<bufSize){ // 1st buf
      buf1[j]=c;
      j++;
    }
    else if(fields==2 && k<bufSize){ // 2nd
      buf2[k]=c;
      k++;
    }
    else if(fields==3 && l<bufSize){ // 3rd
      buf3[l]=c;
      l++;
    }
    c = line[i];
    i++;
  }
  return fields; // Return number of fields counted
}
```

Question 5: C Programming

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
// Struct with 3 fields requested
struct STUDENT{
  char name[50];
  int age;
  double gpa;
};

int main(int argc, char* argv[]){
  // Prompt user
  printf("Enter the size of the array: \n");
  int n; // To store input
  int i; // Counter for students
  char garbage; // To collect CR
  scanf("%d",&n); // Scan input into n
  scanf("%c",&garbage); // Collect CR
  struct STUDENT data[n]; // Array of n students
  int o = 0; // Use o to ask for options
  while(o!=4){//Loop until (4)Quit
    printf("(1) Add a student \n"
           "(2) Delete a student\n"
           "(3) Save all students\n"
           "(4) Quit\n");
    scanf("%d",&o);
    scanf("%c",&garbage); // Collect CR
    if(o == 1){ // Add a student
      printf("Name: \n");
      char name[50];
      scanf("%s",name);
      scanf("%c",&garbage); // Collect CR
      printf("Age: \n");
      int age;
      scanf("%d",&age);
      scanf("%c",&garbage); // Collect CR
```

```
    printf("GPA: \n");
    double gpa;
    scanf("%lf",&gpa);
    scanf("%c",&garbage); // Collect CR
    int t = n; // Temp var for condition
    while(t>=n||t<0){ // While index is invalid
      printf("What cell should they be placed in?\n");
      scanf("%d",&t);
      scanf("%c",&garbage); // Collect CR
      if(t>=n||t<0){
        printf("Invalid cell number! \n");
      }
    }
    // Assign all data
    strcpy(data[t].name,name);
    data[t].age = age;
    data[t].gpa = gpa;
  }
  else if(o == 2){// Delete
    // Could have used a function here
    int t = n;
    while(t>=n||t<0){
      printf("What cell do you want to delete?\n");
      scanf("%d",&t);
      scanf("%c",&garbage); // Collect CR
      if(t>=n||t<0){
        printf("Invalid cell number! \n");
      }
    }
    for(int i=0; i<50; i++){ // Set name to null
      data[t].name[i] = '\0';
    }
    data[t].age = 0;
    data[t].gpa = 0;
  }
  else if(o == 3){// Save
```

```
      int save; // Save or not?
      // Check if file exists
      FILE* in = fopen("students.csv","rt");
      if(in!=NULL){
        printf("File exists already, overwrite?(y/n)\n");
        char ans;
        scanf("%c",&ans);
        scanf("%c",&garbage); // Collect CR
        if(ans == 'y'){
          save = 1; // True to save
        }
        else{
          save = 0; // Don't save
        }
      fclose(in); // Close the reader
      }
      else{
        save = 1; // Overwrite if nothing found
      }
      if(save == 1){ // If we want to save
        FILE* out = fopen("students.csv","wt");
        for(int i=0; i<n; i++){ // Loop through all cells
          if (isalpha(data[i].name[0])){ // Name starts with letter
            fprintf(out,"%s,%d,%1.1lf \n",data[i].name, data[i].age,
                    data[i].gpa); // Write fields
          }
        }
        fclose(out); // Close out
      }

    }
  }
}
```

Question 6: Python

a)

```
file = open("super_villains.txt") # opens file for reading
# missing "r" mode?

name_list = [] # Makes an empty list
for line in file: # Loops through lines of file
    line = line.strip() # Gets rid of CR
    name_list.append(line) # Appends line to list

file.close() # Closes file

i = 0 # counter
# Loops until counter is the length of list or find Morgiana
# Linear search
while i < len(name_list) and name_list[i] != "Morgiana the Shrew":
    i += 1 # Increment counter
if i == len(name_list): # If out of loop and reached end
    print( "No." ) # Did not find Morgiana
else:
    print( "Yes = ",i) # Otherwise, found Morgiana

element = "Joker"; # Set element to joker
low = 0 # Lower bound of list to search
up = len(name_list)-1 # Higher bound, end of list
ff = False # Boolean of whether we found Joker
# Binary search until low>up or we find Joker
while low <= up and ff == False:
    pos = (low + up) / 2 # Mid point
    if name_list[pos] < element: #Mid point less than Joker
        low = pos + 1 # Search upper half
    elif name_list[pos] > element: #Mid point more than Joker
        up = pos - 1 # Search lower half
    else:
        ff = True #Mid point is Joker, true

if ff: # If found Joker
    print(pos) # Print where we found Joker
```

b) Makes a list of all super villains from the provided text file. Checks if Morgiana the Shrew is in the list using a linear search (then prints its position) and then uses binary search (assuming the list is sorted alphabetically) to look for Joker and then prints its position if found.

c) No. 1.
List should look like: C J L M P R