Questions:

To the point questions:

1. what do you understand by term “memory leak”, explain with the help of sample code;
2. What are the key differences between malloc() and calloc()? Explain with the help of code and memory map.
3. What are the main differences between malloc() and “new”? How can you implement two dimensional arrays using new operator in C++?
4. Explain the working of Kernel Level I/O with the help of diagram.
5. Explain the working of Buffered I/O with the help of diagram.
6. When and why Kernel level I/O will be preferred over buffered I/O?
7. When and why buffered I/O will be preferred over kernel level I/O?
8. What fflush and fsync functions do?
9. Explain the working of kernel function pwrite().
10. Describe “flag” and “mode” arguments used in kernel function “open()”.
11. What is partial writing and why it can happen. Provide code to deal with partial writes.
12. What do you understand by non-blocking I/O. In what scenarios it can be used?
13. How realloc() can be used to resize allocated memory in heap. How programmers can get the same effect with “new” operator.
14. Why is the socket function listen(int count) so important for servers? Discuss the impact of variable count.
15. Why are joinable threads preferred over detachable threads? In what conditions detachable threads can be used?
16. why is connection oriented communication preferred for file transfers?
17. What are the key benefits of using connectionless protocols for live streaming?
18. Differentiate asynchronous and synchronous socket functions with examples.
19. How socket functions accept() and connecT() are working in collaboration? Show their working with the help of a 3-way handshake.
20. How socket functions send()A and recv() are working in collaboration? How TCP timers, sequence and acknowledge numbers are used behind these functions? Why must programmers make sure their programs have received all sent data?
21. what is the key difference between File and file descriptor?
22. what advantages we can get to open a file with kernel function “open” instead of “fopen”
23. Describe flag and mode arguments used in kernel function “open()”
24. what is partial writing and why it can happen. Provide code to deal with partial writes.

Question2:

Provide the description of the following I/O functions with the help of example code. Also provide possible examples: (16 points 4 each)

1. read 2. fwrite 3. lseek 4. realloc

In what cases following socket functions will fail and return some error and how that error can be handled? (8points 2 each)

1. int bind(int sid, struct sockaddr \*addrptr, int len)
2. int connect(int sid, struct sockaddr \*addrptr, int len)
3. int accept(int sid, struct sockaddr \*addrptr, int len)
4. int sendto(int sid, const char \*buffer, int len, int flag struct sockaddr \*addrptr, int addrptr len)

Explain the working of the following with the help of detailed diagrams (30 points 10 each)

1. Kernel I/O
2. standard buffered I/O
3. remote procedure call

Question3:

1. Write a complete program in c/c++ to split a file into two files. The source file contains different types of structures given below. Generate two output files as Line.txt and Circle.txt. (8 points)
2. Write a program that can find and show all IP addresses against any domain name resolved through DNS.
3. Write a concurrent server that can handle multiple clients simultaneously. Server is just required to send “hello” message.
4. write a program that can split a file into n equal parts and then join all parts into one file. provide coding in functions splitfile() and joinfile().
5. write a program to demonstrate your skills in RPC, RMI or .NEt remoting. Write and invoke a function to add two numbers and return their sum.

PART 1:

Question 1: T or F

1. The entire process will be block if any one of its kernel level thread call I/O instruction. [ ]
2. Kernel I/O needs more memory to process files than buffered I/O. [ ]
3. Memory allocated through malloc() can be released through delete function. [ ]
4. Kernel I/O function create() calls another kernel function open() to create a file. [ ]
5. fseek() and lseek() both returns file position after successful completion. [ ]
6. SOCK\_STREAM is used to represent TCP in socket programming [ ]
7. vfork() makes a complete copy of the calling process’s address space and allows both the parent and child to execute independently. [ ]
8. In POSIX threads, a thread by default starts as joinable thread. [ ]

Question 2: MCQs

1. “O\_LARGEFILE” is used to open field larger than \_\_\_\_\_GB in size.

a. 1Gb | b. 2Gb | c. 512MB | d 128MB

1. On a UNIX/Linux system everything is a file; if something is not a file it is a \_\_\_\_\_\_\_\_.

( a. device | b. Directory | c. Process | d. option a and b

1. The mask for permission S\_IRUSR | S\_IXGRP | S\_IRWXO will be\_\_\_\_\_. 0417 | b. 0777 c.| 0666 | d. 07144.
2. Which function is used to change allocated memory size in heap. (a. new | b. inc\_malloc() | c. realloc() d.calloc())
3. System call that is used to generate clone of the calling process in unix/linux. (a. system() | b. fork() | c. exec() | d. spawn()
4. which partition is used in UNIX/LINUX for virtual memory. (a. root b. boot c. swap d. /)
5. Joinable thread becomes a zombie after finishing, and won’t disappear until invoke pthread \_\_\_\_\_. (a. join() b. detach() c. exit() d. end()
6. In socket programming which function is used to reserve PORT number. (a. accept b. bind c. socket d. listen)
7. In socket programming which function is used by server to wait for a client request. a. accept b. bind c. socket d. listen
8. which function is used to change alliocated memory size in heap. a. new b. inc\_malloc() c. realloc() d. calloc()

Question 3: FIll in the blanks

1. pwrite() function is used for \_positional\_\_\_ writing.
2. In open() filing function, ) \_\_\_\_ flag is used to rewrite a file and discard all previous data.
3. File \_\_\_\_\_\_ remembers its position in a file and position can be changed without reading or writing in a file by using \_\_\_\_\_\_\_ function. \_\_\_\_\_\_\_\_ argument is used to reposition it from the beginning of the file.
4. \_\_\_\_\_\_\_\_\_\_ function is used to get detailed information (ls -l) of a file.
5. In standard I/O, \_IOF?Bf is used set \_\_\_\_\_\_\_
6. In fopen(), “\_\_\_\_\_ “ is used to open a file in binary append mode.
7. If a file descriptor is available then \_\_\_\_\_\_\_ function is used instead of fopen() to open a given file in buffered I/O.
8. \_\_\_\_\_ function is used to close all files before the ending of a program.
9. \_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_ are used to identify communication sessions at each socket end.
10. SOCK\_STREAM is used to represent \_\_\_\_\_\_\_\_\_\_ transmission layer protocol.
11. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ socket function is used to request an operating system to reserve some network resources.
12. \_\_\_\_\_\_\_\_ and \_\_\_\_\_\_ functions are involved behind TCP 3-way handshake.
13. \_\_\_\_ client is called behind the gethostbyname() function.
14. SOCK\_ \_\_\_\_\_\_\_\_\_ can be used for all kinds of transmission/communications.
15. \_\_\_\_\_\_ function is used to limit number of concurrent socket connections
16. In the open() filing function, O \_\_\_\_\_\_\_\_\_ flag is to rewrite a file and discard all previous data.
17. File \_\_\_\_ remmebrs its position in a file and position can be changed without reading or writing in a file by using \_\_\_\_\_\_\_ function. \_\_\_\_\_\_ argument is used to reposition it from the beginning of the file.
18. \_\_\_\_\_\_ function is used to get detailed information (ls -l) of a file.
19. in sockets \_\_\_\_\_ and \_\_\_\_\_ functions are involved behind full duplex data transfer.
20. SOCK\_\_\_\_\_\_ can be used only for one to one direct communications.
21. In POSIX threads pthread \_\_\_\_ is used to kill thread externally and pthread \_\_\_\_ is used to return value from a thread normally.
22. THe current version of POSIX thread is \_\_\_\_\_\_.

