Section A:

1. Make a program for Executing a child process from parent process.
   what is zombie process? How can you overcome ? Explain through program.
   If Parent execute first then child, what happen? explain through program.
2. What is the output of the following code?

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    if (fork() || fork())
 fork();
    printf("1 ");
    return 0;
}
```

4. What is the output of following code?

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    if (fork()) {
 if (!fork()) {
     fork();
     printf("1 ");
 }
 else {
     printf("2 ");
 }
    }
    else {
 printf("3 ");
    }
    printf("4 ");
    return 0;
}
```

5. What is copy on write? How fork uses it? What is the benefit of using it?

6. Make a program where parent and child processes simultaneously works. Parent will write capital 'ABC D..' into file & child process will write small 'abcd ..' into the file.

Section B:

1. Create 5 children processes from a common parent and ensure that the parent terminates after cleaning all the terminated children using waitpid(). The waitpid() must be called after all the children are created and the parent has completed its work real work,if any. You must interpret the exit code of the cleaned-up processes, using waitpid(), in parent process – you must cover all the possible scenarios !!!

2. Create 5 processes but not from the common parent. Meaning, each child creates a new process. clean-up the children using waitpid(). waitpid() must be called after all the children are created and the parent has completed its work real work,if any. You must interpret the exit code of the

cleaned-up processes, using waitpid(), in the parent process
– you must cover all the possible scenarios !!!

3. Create a 5 child process from a common parent and launch different applications from the children processes - you may launch firefox Gedit, and gcc - parent process must use waitpid() to collect the termination status of the child process - waitpid() must be called after all the children are created and the parent has completed its work real work,if any. You must interpret the exit code of the cleaned-up processes, using waitpid(), in the parent process – you must cover all the possible scenarios - meaning, terminate the children processes normally(successful/ unsuccessful) or abnormally !!!

4. Create 5 processes from a common parent and ensure that the parent terminates after cleaning all the terminated children using waitpid(). The waitpid() must be called after all the children are created and the parent has completed its work real work,if any; in addition, you must compile 5 different program files to in children processes to generate their respective object files ; the parent process must use waitpid() to collect the termination status of children processes – based on the exit code information generated by children processes, link all the object files to generate the final, linked program/application. In addition, the generated binary executable must be loaded in a new process, if the linking is successful

Section C: (Signals)

1. Run the firefox with highest priority.

2. Run the firefox with lowest priority.

3. Change the running firefox process priority to 10

4. Write a program and do the following :using sigaction API, handle SIGINT, SIGTERM, SIGQUIT, SIGSTOP,SIGTSTP and SIGKILL. Install dummy handlers for the mentioned signals Using sigaction() system call API !!!

5. Try to kill init process (with pid 1) from your command line(using kill command) or using kill() system call inside one of your processes.what is the result ? comment on the same.