**Q1.** [10 points] Write a C program that forks as many processes as needed to print your first name. Each process must strictly print one and exactly one character of your name. You should make sure that the characters of your name are printed in the correct order regardless of the order of execution of these processes. For appropriate ordering of printed characters, use **`wait(NULL)`** to make the parent wait for the child to finish. Show your program and a sample compile and run session.

[Hint: by default **`stdout`** is buffered at the line level. Therefore, use "**`setbuf(stdout, NULL)`**" at the beginning of this program to prevent buffering and get the expected output.]

## *Q1 Program*:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/wait.h>
#include <unistd.h>

int main(int argc, char const *argv[]) {
  setbuf(stdout, NULL);
  int length = strlen(argv[1]);
  int k = 0;
  while (k < length) {
    int pid = fork();
    switch (pid) {
    case 0: {
      printf("%c", argv[1][k]);
      return 0;
    }
    default:
      wait(NULL);
    }
    k += 1;
  }
  printf("%s\n", "  ");
}
```

## *Q1 Sample compile and run session:*

**Q2.** [10 points] Write a C program that forks a child. The parent and child use two anonymous pipes to communicate. The parent sends a message to the child. The message is received by the parent program through a command line argument. The child needs to find how many space `' '` characters are there in the input message. The child returns this count to the parent through a pipe, and the parent displays this count by printing a message to the standard output.

- The parent and child should close the unused ends of their pipes
- Assume that the maximum input size is 100 characters

Show your program as well as a sample compile and run session.

**Hints**:

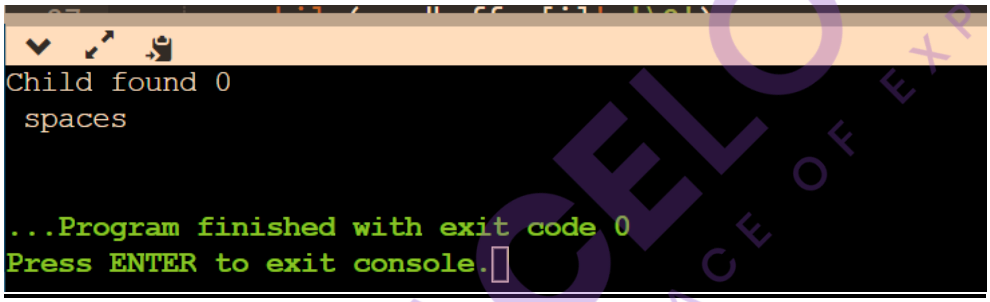
- You can convert the integer count into a char array by using the `sprintf(…)` function.
- See https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm to learn how to use command line arguments in C.

## *Q2 Program*:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char const *argv[]) {
  int childPipe[2], parentPipe[2], nbytes;
  pid_t childpid;
  char readbuffer[100];
  char result[100];
  char message[100];
  char finMessage[100];
  int i = 0;
  int spaces = 0;

  pipe(childPipe);
  pipe(parentPipe);

  if (!fork()) {

    nbytes = read(parentPipe[0], readbuffer, sizeof(readbuffer));
    close(parentPipe[1]);
    close(parentPipe[0]);

    while (readbuffer[i] != '\0') {
      if (readbuffer[i] == ' ') {
        spaces++;
      }
      i++;
    }
    sprintf(finMessage, "%d\n", spaces);
    write(childPipe[1], finMessage, 100);
    close(childPipe[0]);
    close(childPipe[1]);
  } else {
```

```
    write(parentPipe[1], argv[1], 100);
    close(parentPipe[1]);
    close(parentPipe[0]);

    read(childPipe[0], result, sizeof(result));

    close(childPipe[0]);
    close(childPipe[1]);

    // 3. print to screen
    printf("Child found %s spaces\n", result);
  }

  return 0;
}
```

### *Q2 Sample compile and run session:*



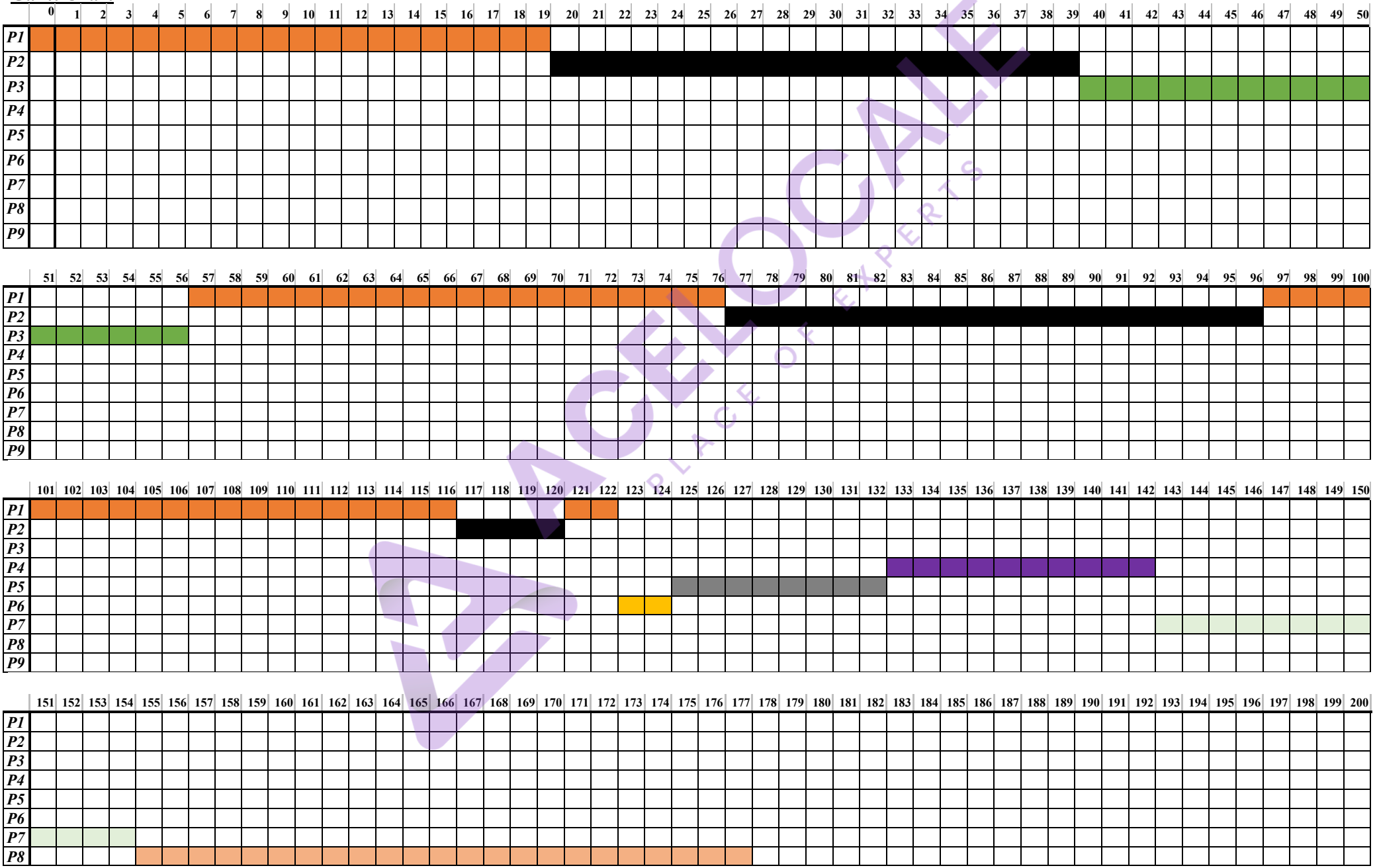
**Q3.** [12 points] A system is using 3-level priorities (levels 0, 1, and 2, where the smaller level the higher the priority). The following algorithms are adopted for each of these levels:

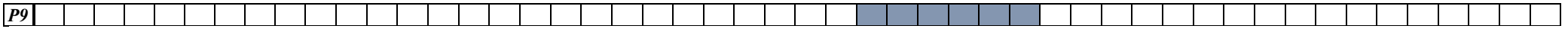
- Level 0: Round robin with quantum time = 20
- Level 1: Preemptive SJF
- Level 2: FCFS

**a.** For the processes shown in the table below show the Gantt chart for scheduling these processes by highlighting the blank Gantt chart below.
**b.** Find the *average waiting time*, *average turnaround time* and *average response time* for this set of processes.

| Process | Arrival Time | Burst Time | Priority Level |
|---|---|---|---|
| ***P1*** | **0** | **62** | **0** |
| ***P2*** | **8** | **44** | **0** |
| ***P3*** | **14** | **17** | **0** |
| ***P4*** | **2** | **10** | **1** |
| ***P5*** | **5** | **8** | **1** |
| ***P6*** | **11** | **2** | **1** |
| ***P7*** | **4** | **12** | **2** |
| ***P8*** | **9** | **23** | **2** |
| ***P9*** | **15** | **6** | **2** |

**Gantt Chart**:






**System Performance Measures**

| | Waiting Time | Response Time | Turnaround Time |
|---|---|---|---|
| **P1** | 0+(57-20)+(97-77)+(121-117) = 61 | 0 | 62+61= 121 |
| **P2** | (20-8) + (77-40) + (117-97) = 69 | 12 | 44+69= 131 |
| **P3** | 40-14 = 26 | 40-14 = 26 | 17+26= 43 |
| **P4** | 133-2 = 131 | 133-2 = 131 | 10+131= 141 |
| **P5** | 125-5 = 120 | 125-5 = 120 | 8+120= 128 |
| **P6** | 123-11 = 112 | 123-11 = 112 | 2+112= 114 |
| **P7** | 143-4 = 139 | 143-4 = 139 | 12+139= 151 |
| **P8** | 155-9 = 146 | 155-9 = 146 | 23+146= 169 |
| **P9** | 178-15 = 163 | 178-15 = 163 | 6+163= 169 |
| **Average** | (61+69+26+131+120+112+139+146+163)/9= 107.4 | 94.3 | 127.89 |