Esercizio 1

Si consideri il seguente codice concorrente:

```
#include <stdio.h>
#include <unistd.h>
int main()
{
   fork();
   fork();
   fork() && fork();

   printf("forked\n");
   return 0;
}
```

Disegnare il relativo albero dei processi, motivandone la costruzione. Si descriva inoltre l'output prodotto dal programma.

Esercizio 2

Si consideri il seguente programma:

```
int main()
{
    pid_t pid;

    fork();
    fork();
    fork();
    pid = fork();
    if (pid)
         printf("Buongiorno\n");
}
```

Si descriva l'albero dei processi e si dica quante volte viene stampata la stringa "**Buongiorno**".

Esercizio 3

Si consideri il seguente segmento di codice concorrente:

```
…
pid_t res;
int a = 1;
int b = 2;

res = fork();
if (res < 0)  {
    perror("fork failed");
    exit(-1); }
a++;
b- -;
if (res == 0) {a++;}
else          {b = 5;}
printf("%d",a);
printf("%d",b);
…
```

Si descriva un possibile output del programma, motivando la risposta data.

Esercizio 4

Descrivere cosa accade quando viene eseguito il codice seguente. Che cosa viene stampato? Sostituire con i termini opportuni il simbolo "***" nelle due **printf()**.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
void conccur()
{
  int ret;
  ret = fork();
  if (ret == 0) {
     printf("\n [%d] Sono il *** \n", getpid());
     while(1);  }
  else          {
     printf("\n [%d] Sono il *** \n", getpid());
     exit(0);   }
}
int main ()
{
  concurr();
  return 0;
}
```

Motivare dettagliatamente la risposta fornita.

Esercizio 5

Il processo P esegue tre system call *fork*() consecutive e quindi termina la propria esecuzione (sincronizzandosi con i propri figli). Quanti processi figli sono stati creati e completati al termine dell'esecuzione di P?

(a) 3
(b) 4
(c) 7
(d) 8

Motivare la risposta data.

Esercizio 6

Descrivere tutti i possibili output prodotti dal programma:

```
#include <stdio.h>
int num = 0;
int main(int argc, char *argv[]){
    int pid;
    pid = fork();
    printf("%d", num);
    if(pid == 0){
        num = 1;
    }else if(pid > 0){
        num = 2;
    }
    printf("%d", num);
}
```

giustificando la risposta fornita.

Esercizio 7

Si consideri il seguente programma, il cui codice è salvato nel file **prova.c**:

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

int main(int argc, char **argv)
{
    printf("Inizio del programma\n");
    int n, conta = 0;
    pid_t pid = fork();

    n = atoi(*++argv);
    if (pid == 0)
    {
        int i = 0;
        for (; i < n; ++i)
            printf("processo ***: conta=%d\n", ++conta);
    }
    else if (pid > 0)
    {
        int j = 0;
        for (; j < n; ++j)
            printf("processo °°°: conta=%d\n", ++conta);
    else
    {
        printf("fork() fallita!\n");
        return 1;
    }
    printf("Fine del programma\n");
    return 0;
}
```

Si descriva un possibile ouput ottenuto a fronte dell'invocazione dei comandi:

```
$ gcc –o prova.c prova
$ chmod u+x prova
$ prova 3
```

definendo anche il significato delle tre linee di comando e l'ovvio contenuto delle stringhe fittiziamente indicate con "***" e "°°°".

Esercizio 8

Scrivere un programma C, che usi la chiamata di sistema **fork()** per generare la successione 1,1,2,6,24,120… (dei numeri fattoriali) all'interno del processo figlio. Il processo figlio produrrà anche le relative stampe. Il genitore dovrà rimanere in attesa, tramite **wait()**, fino alla terminazione del figlio. Il numero di termini da generare sarà specificato a riga di comando (utilizzando opportunamente gli argomenti di **main()**). Implementare i necessari controlli per garantire che il valore in ingresso sia un numero intero non negativo.

Esercizio 9

Nel seguente codice si supponga che:

- Tutte le chiamate di sistema **`fork`** e **`execvp`** vengano eseguite con successo;
- Il codice contenuto nelle diverse **`execvp`** non sia tale da creare nuovi processi o produrre stampe;
- Tutte le variabili **`pid`** siano preventivamente inizializzate a 0.

Quanti processi si creano eseguendo questo codice? Quanti caratteri "**`A`**" e "**`J`**" vengono stampati?

```
void main()
{
      ...
      pid1 = fork();
      printf("A");
      if (pid1) pid2 = fork();
      if (!pid2) {
          pid3 = fork();
          printf("B");
      }
      else {
          execvp(...);
          printf("C");
      }
      if (pid1 != pid3) {
          printf("D");
          if (pid1 && pid3) {
              printf("E");
              execvp(...);
          }
          pid4 = fork();
          printf("F");
      }
      else {
          printf("G");
          pid5 = fork();
      }
      if (pid2) {
          pid6 = fork();
          pid7 = fork();
          if (pid6) pid8 = fork();
          printf("H");
      }
      if (!pid1 && !pid2 && !pid3 && !pid4 && !pid5) printf("I");
      execvp(...);
      printf("J");
}
```