a)

```
int main (int argc, char ** argv)
{
    // Verifica os argumentos passados
    if (argc != 3 [redacted] || atoi(argv[2]) == 0)
        exit(EXIT_FAILURE);

    if (fork() != 0)          // Passa o processo para background
        exit(0);              // libertando a consola
    // Caso o FIFO já exista então o Programa já está a correr
    if (access("fifoCracker", F_OK) == 0)
    {
        Printf("favor esperar - hacking a decorrer");
        exit(0);
    }

    ...
}
```

b)

```
// Cria o FIFO
if (mkfifo("fifoCracker", 0777) != 0)
{
    printf("Erro a criar FIFO");
    exit(EXIT_FAILURE);
}
```

---

```
int filhos[27]; | int i;
for(i = 0; i < 27; i++)
{
    if((filhos[i] = fork()) == 0)        // Cria 27 filhos a executar
    [redacted]                           // o Programa testa
    {
        execl("testa", "testa", NULL);
    }
}
```

```
int i, fifo;
char fifoNome[10];

struct mensagem
{
   char letra;                    // Estrutura da mensagem
   char password[10];
};

struct mensagem mens;

strcpy(mens.password, argv[1]);   // Copia a password

for(i=0; i<27; i++)
{
   sprintf(fifoNome, "fifo%d", filho[i]);  // Cria o nome do fifo
   fifo = open(fifoNome, O_WRONLY);  // Abre o FIFO
   mens.letra = 'a' + i;
   write(fifo, &mens, sizeof(mens));  // Escreve no FIFO
   close(fifo);  // Fecha o FIFO
}
```

[redacted]

```
// O Processo recebe um sinal do tipo SIGUSR2
// quando um filho descobriu a password
signal(SIGUSR2, passwordDescoberta)
Pause();  // Espera pela recepção de um sinal
```

c)

```
void passwordDescoberta(int sig)
{
    char password[10];
    int fifo, dadosRec, i;

    fifo = open("fifoCracker", O_RDONLY);

    do {                    // Lê a password do FIFO
      dadosRec = read(fifo, password, sizeof(password));
    } while (dadosRec > 0);
    close(fifo);

    for (i=0; i<27; i++)
    {
      kill(filhos[i], SIGUSR1);
      // O array filhos foi criado na alínea b)
      // Envia o sinal a todos os filhos
    }

    Printf("%s", Password);
    unlink("fifoCracker");  // Elimina o FIFO
}
```

d)

```
signal(SIGALARM, terminaTimeout); // Associa o sinal alarm à função
alarm(atoi(argv[2]));  // Cria o mecanismo temporizador
                       // com o tempo passado no argumento

void terminaTimeout(int sig) {
    // Operações feitas na alínea anterior
    Printf("timeout");
}
```

e)

```
struct mensagem {
    char letra;              // Estrutura da mensagem
    char Password[10];
};

int main()
{
  int fifo, dadosRec
  struct mensagem mens;
  char password[10], nomeFifo[10];

  sprintf(nomeFifo, "teste %d", getpid()); // Cria o nome do Fifo
  mkfifo(nomeFifo, 0777);  // Cria o Fifo
  fifo = open(nomeFifo, O_RDONLY); // Abre o Fifo

  do {
     dadosRec = read(fifo, &mens, sizeof(mens));
                                   // Lê do Fifo
  } while(dadosRec > 0);
  close(fifo);

  password = descobre(&(mens.letra)); // Descobre o password
  if (Password == NULL)
     exit(0);
  else {
     kill(getppid(), SIGUSR2); // Envia o sinal ao "servidor"
     fifo = open("fifoCracker", O_WRONLY); // Abre o fifo
     write(fifo, Password, sizeof(Password)); // Escreve no Fifo
     close(fifo);
  }
  unlink(nomeFifo); // Apaga o seu Fifo
  exit(0);
}
```

f)
```
...
signal (SIGUSR1, terminaOrdeiramente);
// Associação do sinal
...

void terminaOrdeiramente (int sig)
{
    char nomeFifo[10];
    sprintf(nomeFifo, "testa%d", getpid());
    unlink(nomeFifo);
    exit(0);
}
```