```c
#include <stdio.h>
unsigned long fact_term(unsigned long n, unsigned long s) {
  if(n < 2) return s;
  return fact_term(n - 1, n * s);
}
unsigned long fib(unsigned long n) {
  if(n < 1) return 0;
  if(n == 1 || n == 2) return 1;
  return fib(n - 1) + fib(n - 2);
}
int main(void) {
  unsigned long i;
  for(i = 0; i <= 12; i++)
        printf("%lu! = %lu\n", i, fact_term(i, 1));
  for(i = 0; i <= 47; i++)
        printf("fib(%lu) = %lu\n", i, fib(i, 1, 1));
  return 0;
}
```

FIG. 1 – Factorielle en récursivité terminale et Fibonacci en récursivité simple (`factnfib.c`).

```c
#include <stdio.h>
int main(void) {
  unsigned char c = 'A', i;
  for (i = 128; i != 0; i = i / 2)
    if (c & i) putchar('1');
    else putchar('0');
  putchar('\n');
  return 0;
}
```

```c
#include <stdio.h>        /* Pour les fonctions printf, fprintf et putchar */
#include <stdlib.h>       /* Pour la fonction atol, strtoul ... */
int main(int argc, char ** argv) {
  unsigned long v = 0, i, j;
  if(argc != 2) {
    fprintf(stderr, "usage : %s <nombre a convertir>\n", argv[0]);
    exit(1);
  }
  j = 1UL << ( (sizeof(v) << 3) - 1);
  v = strtoul(argv[1], NULL, 10);
  printf("Le nombre %lu s'ecrit : ", v);
  for (i = 0; i < (sizeof(v) << 3); i++, j >>= 1) {
    if (v & j) putchar('1');
    else putchar('0');
  }
  printf(" en binaire\n");
  return 0;
}
```

FIG. 2 – Conversion d'un décimal en binaire (`dec2bin.c`).

Devoir 02 :

- Écrire une fonction itérative pour calculer Factorielle ;
- Écrire une fonction itérative pour calculer Fibonacci ;
- Écrire les deux fonctions `somme_i` (itérative) et `somme_rt` (récursivité terminale) qui calculent la somme des n premiers entiers ;
- Écrire la fonction `fib_rt` qui calcule Fibonacci en récursivité terminale ;
- Modifier le `main` de `fact.c` (cf. FIG.4) afin de pouvoir calculer plusieurs factorielles sans relancer le programme ;
- Modifier `dec2bin.c` (cf. FIG.2) afin d'obtenir une impression en base octale (8) ;
- Modifier `dec2bin.c` (cf. FIG.2) afin d'obtenir une impression en base hexadécimale (16) ;
- Envoyer l'ensemble (archivé en : `zip, tar, tar.gz ou .tgz`) par email.

```
1  # Makefile
2
3  CC=gcc
4  CFLAGS=-Wall -O2
5  LDFLAGS=
6  SRC=dec2bin.c
7  OBJ=$(SRC:.c=.o)
8  PROG_NAME=dec2bin
9
10 # cibles et dependances
11
12 all: $(PROG_NAME)
13
14 %.o: %.c
15         $(CC) $(CFLAGS) -c $<
16
17 $(PROG_NAME) : $(OBJ)
18         $(CC) $(LDFLAGS) $^ -o $@
19
20 clean:
21         rm -f $(PROG_NAME) $(OBJ) *~
```

FIG. 3 – Exemple de Makefile pour `dec2bin.c`.

```
1  #include <stdio.h>  /* pour printf et scanf */
2  #include <stdlib.h> /* pour atol            */
3  #include <ctype.h>  /* pour isdigit         */
4  unsigned long fact(unsigned long n) {
5    if(n > 1)
6      return n * fact(n - 1);
7    return 1;
8  }
9  int main(void) {
10   unsigned long n;
11   printf("Entrez un nombre\n");
12   if(scanf("%lu", &n) == 1)
13     printf("Fact(%lu) = %lu\n", n, fact(n));
14   return 0;
15 }
16 /*** OU BIEN ***
17 int main(void) {
18   unsigned char buff[50], c, i = 0;
19   while( (c = getchar()) != '\n') {
20     if(isdigit(c))
21       buff[i++] = c;
22     else
23       break;
24   }
25   buff[i] = '\0';
26   printf("Fact(%l) = %l\n", atol(buff), fact(atol(buff)));
27   return 0;
28 }*/
```

FIG. 4 – Fonction récursive pour le calcul de la factorielle `fact.c`.