SCHRECK
Corentin

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

//exo 10
double fraction(int a,int b)
{
   double fa=a;
   double fb=b;
   double f=fa/fb;
   return(f);
}

//exo 11
bool bissextile(int year)
{
   if(year%400==0) return(true);
   if(year%100==0) return(false);
   if(year%4==0) return(true);
   return(false);
}

//exo 12
double evalue(double coefs[],int size, double x)
{
   double result=coefs[size-1];
   for (int i=1;i<size;i+=1)
   {
      result*=x;
      result+=coefs[size-i-1];
   }
   return(result);
}

//exo 13
int ecart_min(int tab[], int size)
{
   int e_min=abs(tab[0]-tab[1]);
   for (int i=1;i<size;i+=1)
   {
      if (abs(tab[i-1]-tab[i])<e_min) e_min=abs(tab[i-1]-tab[i]);
   }
   return(e_min);
}
```

SCHRECK
Corentin

```
//exo 14
int len(char string[])
{
   int i=0;
   for (i;string[i]!='\0';i+=1);
   return(i);
}
int intfromstring(char string[], int n, bool i, int* pn)
{
   if (n<len(string))
   {
      int nb=0;
      while(48<=(int)string[n] && (int)string[n]<=57)
      {
         nb*=10;
         nb+=((int) string[n])-48;
         n+=1;
      }
      if (i) *pn=n;
      return(nb);
   }
   return(0);

}
int extraire(char string[])
{
   int n=0;
   bool intnotfound=true;
   while (intnotfound && (int)string[n]!=0)
   {
      if (string[n]=='-')
      {
         return(-intfromstring(string,n+1,false,&n));
      }
      if (48<=(int)string[n] && (int)string[n]<=57)
      {
         return(intfromstring(string,n,false,&n));
      }
      n+=1;


   }
   return(0);

}
```

SCHRECK
Corentin

```
//exo 15
double sommemajo(double tab[],int n)
{
   int nb_p=0;
   int nb_n=0;
   double result_p=0;
   double result_n=0;
   for(int i=0;i<n;i+=1)
   {
      if (tab[i]>0)
      {
         result_p+=tab[i];
         nb_p+=1;
      }
      if (tab[i]<0)
      {
         result_n+=tab[i];
         nb_n+=1;
      }
   }
   if (nb_p>nb_n)
   {
      return(result_p);
   }
   if (nb_n>nb_p)
   {
      return(result_n);
   }
   return(0);
}

//exo 16 (on supposera que seuls les 95 caractères usuels de la table ASCII sont présents dans la chaîne)
bool charinstring(char c,char string[])
{
   int i=0;
   bool in=false;
   int l = len(string);
   while (i<l && !in)
   {
      if (string[i]==c) in=true;
      i+=1;
   }
   return(in);
}
```

SCHRECK
Corentin

```
int nbdiff(char string[])
{
   int nb=0;
   char seen_chars[96];
   seen_chars[0]='\0';
   int i=0;
   int l=len(string);
   while(i<l && nb<95)
   {
      if(!charinstring(string[i],seen_chars))
      {
         seen_chars[nb]=string[i];
         nb+=1;
         seen_chars[nb]='\0';
      }
      i+=1;
   }
   return(nb);
}

//exo 17
int power(int a,int b)
{
   if (b==0) return(1);
   if (b==1) return(a);
   if (b%2==0) return(power(a*a,b/2));
   return(a*power(a*a,(b-1)/2));
}
bool palindromeint(int nb)
{
   int power_10=0;
   int holder=nb;
   while (holder>=10)
   {
      holder/=10;
      power_10+=1;
   }
   int i=0;
   while(i<=power_10/2)
   {
      int i_digit=(nb/power(10,i))-10*(nb/power(10,i+1));
      int p_digit=(nb/power(10,power_10-i))-10*(nb/power(10,power_10-i+1));
      if (i_digit!=p_digit)
      {
         return(false);
      }
      i+=1;
   }
   return(true);
}
```

SCHRECK
Corentin

```
//exo 18
bool palindrome2(char string[])
{
   int l=len(string);
   char* buffer=malloc(l*sizeof(char));
   buffer[0]='\0';
   int j=0;
   for (int i=0;i<l;i+=1)
   {
      int c=(int) string[i];
      if (97<=c && c<=122)
      {
         buffer[j]=(char) c;
         j+=1;
         buffer[j]='\0';
      }

      if (65<=c && c<=90)
      {
         buffer[j]=(char) c+32;
         j+=1;
         buffer[j]='\0';
      }
   }
   for (int i=0;i<(j-1)/2;i+=1)
   {
      if (buffer[i]!=buffer[j-1-i])
      {
         free(buffer);
         return(false);
      }
   }
   free(buffer);
   return(true);

}
```

SCHRECK
Corentin

```
//exo 19
int sommeentiers(char string[])
{
   int l=len(string);
   int s=0;
   int i=0;
   while(i<l)
   {
      if (string[i]=='-')
      {
         s-=intfromstring(string,i+1,true,&i);
      }

      if (48<=(int)string[i] && (int)string[i]<=57)
      {
         s+=intfromstring(string,i,true,&i);
      }
      i+=1;
   }
   return(s);
}

int main()
{
   printf("[exo 10] \n");
   int a=5;
   int b=2;
   printf("a/b=%f \n",fraction(a,b));

   printf("\n[exo 11] \n");
   printf("2042 est bissextile : %d \n",bissextile(2042));
   printf("2420 est bissextile : %d \n",bissextile(2420));
   printf("2100 est bissextile : %d \n",bissextile(2100));
   printf("2400 est bissextile : %d \n",bissextile(2400));

   printf("\n[exo 12] \n");
   double x=42.0;
   double deg_0[]={5.0};
   double deg_1[]={0.0,10.0};
   double deg_2[]={8.0,4.0,-2.0};
   printf("f(x)=5 evaluee en 42 : %f \n",evalue(deg_0,1,x));
   printf("f(x)=10x+0 evaluee en 42 : %f \n",evalue(deg_1,2,x));
   printf("f(x)=-2x^2+4x+8 evaluee en 42 : %f \n",evalue(deg_2,3,x));

   printf("\n[exo 13] \n");
   int tab[]={5,3,6,7,1,2};
   printf("ecart minimum entre deux elements consecutifs de tab : %d \n",ecart_min(tab,6));

   printf("\n[exo 14] \n");
   printf("le premier entier dans la chaine \"lorem ipsum 42\" est : %d",extraire("lorem ipsum 42"));
   printf("le premier entier dans la chaine \"lorem ipsum -42 42\" est : %d",extraire("lorem ipsum -42 42"));
```

SCHRECK
Corentin

```
    printf("\n[exo 15] \n");
    double tab2[]={-1.0,2.0,5.0,1.0,-25.0};
    printf("la somme des elements majoritaires de tab2 est : %f \n",sommemajo(tab2,5));
    double tab3[]={-1.0,2.0,5.0,-1.0,-25.0};
    printf("la somme des elements majoritaires de tab3 est : %f \n",sommemajo(tab3,5));
    double tab4[]={2.0,5.0,-1.0,-25.0};
    printf("la somme des elements majoritaires de tab3 est : %f \n",sommemajo(tab4,4));

    printf("\n[exo 16] \n");
    char string[]="Hello World !";
    printf("le nombre de caracteres differents dans %s est : %d",string,nbdiff(string));

    printf("\n[exo 17] \n");
    printf("42188124 est un palindrome : %d \n",palindromeint(42188124));
    printf("4218124 est un palindrome : %d \n",palindromeint(4218124));
    printf("4218 est un palindrome : %d \n",palindromeint(4218));

    printf("\n[exo 18] \n");
    char c1[]="LoreM ipSum mUspi meRol";
    printf("\"%s\" est un palindrome : %d \n",c1,palindrome2(c1));
    char c2[]="lore_m  ip_sumus()pi me()rol";
    printf("\"%s\" est un palindrome : %d \n",c2,palindrome2(c2));
    char c3[]="lorem ipsum";
    printf("\"%s\" est un palindrome : %d \n",c3,palindrome2(c3));

    printf("\n[exo 19] \n");
    char c4[]="42 lorem ipsum -18";
    printf("somme des entiers dans \"%s\" : %d \n",c4,sommeentiers(c4));



    return(0);
}
```