1 Programmer une commande cp simplifiée (∼ 14 points)

Question 1 :

1 point

1. `cp ../TP42/PROGS/*.sh .`
2. `cp ../TP42/PROGS/Makefile Makefile42`

Question 2 :

2 points

```
int copie(char nom_src[], char nom_dest[]){
FILE *fd=fopen(nom_dest,"w"), *fs=fopen(nom_src,"r");
char c;

if(fs==NULL) return 1;
if(fd==NULL) return 2;

fscanf(fs,"%c",&c);
while(!feof(fs)){
        fprintf(fd,"%c",c);
        fscanf(fs,"%c",&c);
        }

close(fd);
close(fs);

return 0;
}
```

Question 3 :

1 point

```
int main(int argc, char *argv[]) {
int code ;
if (argc != 3) {
        printf("usage %s fich_source fich_dest\n", argv[0]) ;
        exit(3) ;
        }

code = copie(argv[1], argv[2]) ;

if (code == 1) {
        printf("erreur d'ouverture de %s en lecture \n", argv[1]) ;
        exit(1) ;
        }

if (code == 2) {
        printf("erreur d'ouverture de %s en ecriture \n", argv[2]) ;
        exit(2) ;
        }

return 0 ;
}
```

Question 4 :

1 point

— En utilisant les fonctions de la bibliiothèque **string**

```
int slash_fin(char d[], char s[]) {
int l = strlen(s) ;
strcpy(d, s) ;
if (d[l-1] != '/') {
        d[l]='/';
        d[l+1]='\0';
        l++ ;
        }
return l ;
}
```

— Sans utiliser les fonctions de la bibliiothèque **string**

```
int slash_fin(char d[], char s[]) {
int i =0 ;
while (s[i] != '\0') {
        d[i] = s[i] ;
        i++ ;
        }
if (d[i-1] != '/') {
        d[i]='/' ;
        i++ ;
        }
d[i]='\0' ;
return i ;
}
```

Question 5 :

1 points

Multiples solutions possibles :

```
int indice_apres_dernier_slash(char f[]){
int l ;
int i;
l = strlen(f) ; // ou calcul de la longueur

i=l ;
while (i>=0 && f[i] != '/') {
        i-- ;
        }

return i+1 ;
}
```

ou (un seul parcours) :

```
int indice_apres_dernier_slash(char f[]) {
int i, res ;

res=0 ;
i=0 ;
while (f[i]!='\0') {
        if (f[i]=='/') {
                res=i+1;
                }
```

```
        i++ ;
        }
return res ;
}
```

Question 6 :

2 points

```
void chemin_cible(char c[], char r[], char f[]){
int i, j ;

i = slash_fin(c, r) ;
j = indice_apres_dernier_slash(f) ;
while(f[j] != '\0') {
        c[i]=f[j] ;
        i++ ;
        j++ ;
        }
c[i]='\0';
}
```

Question 7 :

2 points

```
int main(int argc, char *argv[]) {
char CH[100] ;
int code ;
int i ;

if (argc <= 2) {
        printf("usage %s fich_source fich_dest\n", argv[0]) ;
        printf("ou  %s fich_source_1 fich_source_2 ... rep_dest\n", argv[0]) ;
        exit(3) ;
        }

if (argc == 3) {
        code = copie(argv[1], argv[2]) ;
        if (code == 1) {
                printf("erreur d'ouverture de %s en lecture \n", argv[1]) ;
                exit(1) ;
                }
        if (code == 2) {
                printf("erreur d'ouverture de %s en ecriture \n", argv[2]) ;
                exit(2) ;
                }
        }
else {
        for (i=1 ; i<argc-1 ; i++) {
                chemin_cible(CH, argv[argc-1], argv[i]) ;
                code = copie(argv[i], CH) ;
                if (code == 1) {
                        printf("erreur d'ouverture de %s en lecture \n", argv[i]) ;
                        exit(1) ;
                        }
                if (code == 2) {
                        printf("erreur d'ouverture de %s en ecriture \n", CH) ;
                        exit(2) ;
                        }
```

```
                }
        }

return 0 ;
}
```

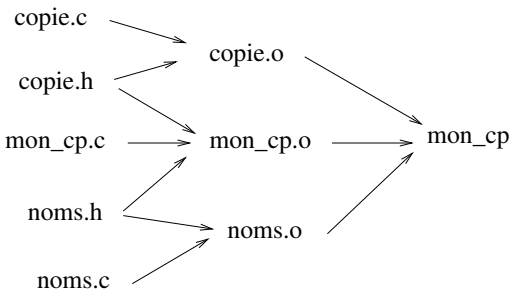
Question 8 :

1 point

1. — **copie.h** :
   ```
   int copie(char nom_src[], char nom_dest[]) ;
   ```
   — **noms.h** :
   ```
   void chemin_cible(char c[], char r[], char f[]);
   ```
2. — **mon_cp.c** :
   ```
   #include "copie.h"
   #include "noms.h"
   ```
   — **copie.c** :
   ```
   #include "copie.h"
   ```
   — **noms.c** :
   ```
   #include "noms.h"
   ```

Question 9 :

3 points




```
CC=gcc
CFLAGS=-Wall -Werror -g
mon_cp: noms.o copie.o mon_cp.o
gcc -o mon_cp noms.o copie.o mon_cp.o

noms.o: noms.h  noms.c
$(CC) $(CFLAGS) -c noms.c

copie.o: copie.h copie.c
$(CC) $(CFLAGS) -c copie.c

mon_cp.o: copie.h noms.h mon_cp.c
$(CC) $(CFLAGS) -c mon_cp.c
```

2 Gérer une corbeille (∼ 6 points)

2.1 Ecriture de nombre_dans_corbeille.sh

Question 10 :

1 point

```
ls ~/Corbeille/ | grep  $1'\..*' | wc -w
```

On accepte aussi :

```
ls ~/Corbeille/$1.* | wc -w
```

2.2 Ecriture de supprime.sh

Question 11 :

5 points

```
#!/bin/bash

if [ $# -lt 1 ]
then
    echo usage $0 fich1 fich2 ...
    exit 1
fi

# On s'assure que ~/Corbeille existe

if [ ! -d ~/Corbeille ]
then
    mkdir ~/Corbeille
fi

# Pour tous les arguments ...
for i in $*
do

    # Si l'argument n'est pas un fichier
    if [ ! -f $i ]
    then
        echo $i n est pas un fichier
    else
    # sinon

        NOM=`basename $i`  # nom sans le chemin

        # on compte le nombre de d'occurence de nom.nombre déja dans
        # la corbeille
        NUM=`./nombre_dans_corbeille.sh $NOM`

        # le nouveau fichier dans la corbeille aura une extension
        # NUM+1
        NUM=`expr $NUM + 1`

        # On déplace en renommant
        mv $i ~/Corbeille/$NOM.$NUM
    fi
done
```

2.3 Ecriture de restaure.sh

Question 12 :

```
#!/bin/bash

# il faut un nom de fichier en argument
if [ $# -ne 1 ]
```

```
then
    echo usage $0 fich
    exit 1
fi

# teste si l'argument ne contient pas de chemin

if [ `basename $1` != $1 ]
then
   echo usage fich ne doit pas contenir de chemin
   exit 3
fi

# on compte combien il y a de sauvegardes existantes de $1

NUM=`./nombre_dans_corbeille.sh $1`

# s'il n'y en a pas, erreur !

if [ $NUM -eq 0 ]
then
    echo $1 n existe pas dans la corbeille
    exit 3
fi

echo restauration $1

# on restaure la plus récente
mv ~/Corbeille/$1.$NUM .
```