INPF12 : projet 2024-2025

1 Dates et principes

Projet à rendre pour le vendredi **07/05/2025** à **23h59**, aucun retard ne sera toléré.
Des soutenances pourront être organisées ensuite.
Lire tout le sujet (tout ? tout).
Un rendu de projet comprend :

— Un rapport précisant et justifiant vos choix (de structures, etc.), les problèmes techniques qui se posent et les solutions trouvées ; il donne en conclusion les limites de votre programme. Le rapport sera de préférence composé avec LaTeX. Le soin apporté à la grammaire et à l'orthographe est largement pris en compte.
— Un manuel d'utilisation, même minimal.
— Un code *abondamment* commenté ; la première partie des commentaires comportera systématiquement les lignes :

1. Le type prévu de la fonction
2. `@requires` décrivant les pré-conditions : c'est-à-dire conditions sur les paramètres pour une bonne utilisation (*pas de typage ici*),
3. `@ensures` décrivant la propriété vraie à la sortie de la fonction lorsque les pré-conditions sont respectées, le cas échéant avec mention des comportements en cas de succès et en cas d'échec,
4. `@raises` énumérant les exceptions éventuellement levées (et précisant dans quel(s) cas elles le sont).

On pourra préciser des informations additionnelles si des techniques particulières méritent d'être mentionnées Le code doit enfin **compiler** sans erreur (évidemment) et sans warning sur les machines des salles de TP.

Si vous avez des questions concernant ce sujet, vous pouvez envoyer, avant le 02/05/2025, un mail à julien.forest@ensiie.fr avec christophe.mouilleron@ensiie.fr en copie.

2 Protocole de dépôt

2.1 Documents à rendre

Vous devez rendre :
— votre rapport au format `PDF`,
— vos fichiers de code,
— vos tests,
rassemblés dans une archive tar gzippée identifiée comme *votre_prénom_votre_nom*.tgz . La commande devrait ressembler à :

```
tar cvfz randolph_carter.tgz rapport.pdf fichiers.ml autres_truc_éventuels
```

Lisez le man et testez le contenu de votre archive. Une commande comme par exemple : `tar tvf randolph_carter.tgz` doit lister les fichiers et donner leurs tailles. Une archive qui ne contient pas les fichiers demandés ne sera pas excusable. Une archive qui n'est pas au bon format ne sera pas considérée.

2.2 Procédure de dépôt

Vous devez enregistrer votre archive tar dans le dépôt `inpf12-projet-2024` en vous connectant à http://exam.ensiie.fr

3 Contexte

On souhaite programmer un éditeur de texte avec des fonctions simples : insérer ou effacer un caractère, se déplacer dans le texte.

La notion principale à manipuler est la notion de buffer. Un buffer est un ensemble de lignes. Un ligne est un ensemble de caratères .

À cette fin, il vous est fourni, dans le même repertoire que ce document, un squelette de code. Ce code comprend :
— une liste de charatères valides pour l'éditeur ;
— un type `zipper` qui permet de coder les lignes et les buffers ;
— la définition du type `line` (zipper avec un curseur, les caractères quui le précèdent et les caratères qui le suivent) ;

— la définition du type `buffer` (zipper avec une ligne courante, les lignes précédentes et les lignes suivantes) ;
— la definition du type `action` permettant de représenter les actions possibles pour l'utilisateur ;
— la gestion de la partie graphique de l'application ;
— la définition plus que partielle d'un certain nombres de fonctions que vous devrer complèter.

On remarquera qu'un buffer contient forcément au moins une ligne.

4 Travail à fournir

4.1 Première partie

Vous commencerez par implanter des fonctions sur les `zipper`, en particulier :
— `get_current : ('a,'b) zipper -> 'b` retournant la valeur courrante du zipper
— `get_pos : ('a,'b) zipper -> int` retournant la position courrante du zipper
— `fold_zipper : ('a -> 'c -> 'c) -> ('b -> 'c -> 'c) -> ('a,'b) zipper -> 'c -> 'c` implantant un fold sur le type des zipper.
— `update_with : ('b -> 'c) -> ('a,'b) zipper -> ('a,'c) zipper` qui applique la fonction passée en argument au champ `current` du zipper et retourne le zipper résultant.

Pour les déplacements, vous pourrez au choix :

1. faire deux fonctions opérant sur le type `line` et deux autres opérant sur le type `buffer`
2. Faire deux fonctions génériques (opérant sur le type `('a,'b) zipper`) en vous inspirant de la fonction `update_with`.

4.2 Deuxième partie

Vous implanterez les fonctions suivantes, nécessaires pour gérer les actions effectuées par l'utilisateur.

— `move_left : buffer -> buffer` effectuant un déplacement vers la gauche de la ligne courrante ;
— `move_right : buffer -> buffer` effectuant un déplacement vers la gauche de la ligne courrante ;
— `move_up : buffer -> buffer` effectuant un déplacement vers le haut dans le fichier ;
— `move_down : buffer -> buffer` effectuant un déplacement vers le haut dans le fichier ;
— `insert_char : char -> buffer -> buffer` ajoutant le `char` à la position courante et avancant la position de 1 ;
— `do_supr : buffer -> buffer` supprimant le caractère courrant.
— `do_backspace : buffer -> buffer` supprimant le caractère précédent le caractère courrant ;
— `create_newline: buffer -> buffer` insérant un saut de ligne à la position courrante.

Dans un premier temps, on conviendra de ne pas effectuer de déplacement :
— vers la gauche si on est en début de ligne ;
— vers la droite si on est en fin de ligne ;
— vers le haut si on est sur la première ligne ;
— vers le bas si on est sur la dernière ligne.

4.3 Troisième partie

Proposer une implantation plus ergonomique des fonctions de la partie deux (meilleurs déplacements, meilleure gestion des suppressions).

5 Fichiers fournis

Le répertoire contenant ce fichier comprend également :
— un fichier de squelette de code `projet.ml` ;
— un Makefile vous permettant de compiler vos projets sur les machines de l'ENSIIE.

Si vous souhaitez travailler sur vos machines personnelles, vous pouvez :
— utiliser le Makefile fourni, en commentant la ligne 6 et en décommentant la ligne 7 ;
— lancer une commande du type

```
ocaml -I +graphics graphics.cma -init projet.ml
```

pour tester en mode interprété sur vos machines,
Pour cela, il vous faudra installer le paquet `libgraphics-ocaml-dev`.

6 Utilisation de l'interface

Les événements clavier et l'affichage seront gérés à l'aide du code déjà fourni et vous n'aurez rien de particulier à faire. En dehors des caractères d'édition normaux, quelques actions spéciales seront possibles :

Touches (simultanées)	Action
Control q	déplacement du curseur d'un caractère vers la gauche
Control d	déplacement du curseur d'un caractère vers la droite
Control z	déplacement du curseur d'un caractère vers le haut
Control s	déplacement du curseur d'un caractère vers le bas
Backspace	effacement du caractère précédent
Delete	effacemet du caractère courant
Escape	arrêt du programme