DEA BBSG. Examen d'Informatique

Décembre 2002

Durée: 2 heures - Documents autorisés Daniel GAUTHERET

1. (6 pt)

Identifiez parmi les commandes Perl suivantes celles qui comportent une erreur. Corrigez l'erreur.

```
a. for ($j = 0 , $j < $completematch , $j-=1) {
b. @hsplines = ($hsp =~ /Query:.+$/gm);
c. $fichconsens='>>'+$fich+'consen.csu';
d. while ($occur = each %nbr_de_sigAATAAA_use) {
e. if ($nb_match eq 2) {push (@good_NONE, "$nb_match,$position")}
f. open REM, ">$seqname.rem" or die "impossible\n";
```

2. (7 pt)

Considérez le programme Perl en Annexe 1

a. Que fait ce programme ?
b. Quelle est la fonction de la partie de code marquée « 1» ?
c. Quelle est la fonction de la partie de code marquée « 2 » ?
d. A quoi servent les variables @DiNt et $DiNtCnt?
e. Quel serait la sortie du programme si on lui fournissait en argument le fichier en annexe 2 ?

3. (3 pt)

Ecrivez un programme Perl lisant un fichier fasta et créant la table de tous les mots de 15 lettres (comme celle utilisée par BLASTN).

4. (4 pt)

a) Ecrivez en Perl une boucle lisant un fichier d'annotation au format suivant et affichant les coordonnées de chaque CDS

```
CDS             262095..262170
CDS             475648..475785
CDS             complement(695887..695963)
```

b) Même question en permettant les CDS multi-exons de la forme:

```
CDS             join (475648..475785, 475800..475880)
```

ANNEXE 1

```
$win=10;
$repstr = 'N' x $win;
$purity = 8;

if (!open(F, $ARGV[0])) {
  print "Can't open \"$ARGV[0]\"\n";
  exit;
}

$line = <F>;
while ($line) {
  if ($line =~ /^>/) {
    $name = $line;
    $seq = "";
    while (($line = <F>) && ($line !~ /^>/)) {
      chop $line;
      $line =~ tr/\s//d;
      $line =~ tr/[a-z]/[A-Z]/;
      $line =~ tr/[tT]/[uU]/;
      $seq .=  $line;
    }
    $seq2 = $seq;
    if (length($seq) >= $win) {
      for ( $i = $win-1 ; $i < length($seq); $i++){
         $seg = substr ($seq, $i-$win+1, $win);
         @DiNt = ($seg =~ /[AUN]/g);
         $DiNtCnt = $#DiNt+1;
         if ($DiNtCnt < $purity) {
           @DiNt = ($seg =~ /[GUN]/g);
           $DiNtCnt = $#DiNt+1;
          if ($DiNtCnt < $purity) {
            @DiNt = ($seg =~ /[CUN]/g);
             $DiNtCnt = $#DiNt+1;
            if ($DiNtCnt < $purity) {
               @DiNt = ($seg =~ /[GAN]/g);
               $DiNtCnt = $#DiNt+1;
               if ($DiNtCnt < $purity) {
                 @DiNt = ($seg =~ /[CAN]/g);
                 $DiNtCnt = $#DiNt+1;
                 if ($DiNtCnt < $purity) {
                  @DiNt = ($seg =~ /[GCN]/g);
                    $DiNtCnt = $#DiNt+1;
                 }
               }
            }
          }
        }
      }
      if ($DiNtCnt >= $purity) {
         $seq2 = substr ($seq2, 0, $i-$win+1) .
                 $repstr . substr ($seq2, $i+1);
      }
    }
    print $name;
    print $seq2;
  } else {
    $line = <F>;
  }
}
```

ANNEXE 2

```
>gene 1
ATGCATATATATATATGCATGCATGCATGCATGCATGC
>gene 2
ATGCATGCGGGGGGGGGGGGGGATGC
```