Zadanie 4

Wojciech Miśta, 236453

Działanie programu:

```
G8Drosophilahybri       TGG___T_TGA____A__A__A__CG____AA___________C_G__GCA_G___GCA     1 - 59
T15Drosophilahybr       TGGACTT_TCA__TCA__A__A__TGTTTTTC___________C_C__GCATA___G_A     1 - 59
G11Drosophilahybr       T_G___TATAATT__AGCAGCAG_CG____AAGCTGCCGCCGCC_GCAGCA_GTT_GGA     1 - 59
C6Drosophilahybri       T_G___TATGA____A__AT_ATTC_____AT___________CTG__GAA_GATATCA     1 - 59
                        * *   * * *    *  *  *                     *    * *      *

G8Drosophilahybri       __T____AA_____C_____G_____T_____TTG_C___CT__T_T_T_GTTA_C__     60 - 118
T15Drosophilahybr       __A____AACTTTTTC_____G_____T___A_TTGTC_ATCT_AT_TCG_GTT_____     60 - 118
G11Drosophilahybr       GGTGAACAA_____C_____GAAACATACAGAATG_CGA_CT__TAT_TCATTT_C_T     60 - 118
C6Drosophilahybri       __TG___AAT_____CATTGAG_____T_____TCA_C___CTC_G_T_T_CCTACCA_     60 - 118
                              **      *     *     *         *   **    *     *

G8Drosophilahybri       C_G___G_GC_GC_A___G_A_____A_______C__A___GA_AT_C__C_G_T__G     119 - 177
T15Drosophilahybr       __G__TG_GCATC_A_TCGAA_TTGTCA_____CC__A___GA_AT____T_GTT__T     119 - 177
G11Drosophilahybr       C_____A_AC_G__A_____AC_____A__TTTT_C__A_TTTACTTCC_ACAGAT__G     119 - 177
C6Drosophilahybri       CTGTT_GCGC__CTAC__A_A_____ATG_____CTTAT__AA_AT_CA_T_G_TAGG     119 - 177
                              *    *     *      *       *  *    *  *      * *

G8Drosophilahybri       __________AGA_GGA_A_A_AC_TCAG_G__GA_A_A_AAG_AT__C____CT_CT_     178 - 236
T15Drosophilahybr       __________ATA___ATC_A_AT_TC_____GATACT_TGGTAT_AC_TTGGTACT_     178 - 236
G11Drosophilahybr       _________CTACACGCATATC_GCGCCAG_GTTTC_A_C_AA__A_________CT_GTA     178 - 236
C6Drosophilahybri       TCTTACAA__AGA_AGA_A_AGA____AGCA__GA_A_AGAA__ATA_CA___CA_C__     178 - 236
                              * *   *                    *      *
```

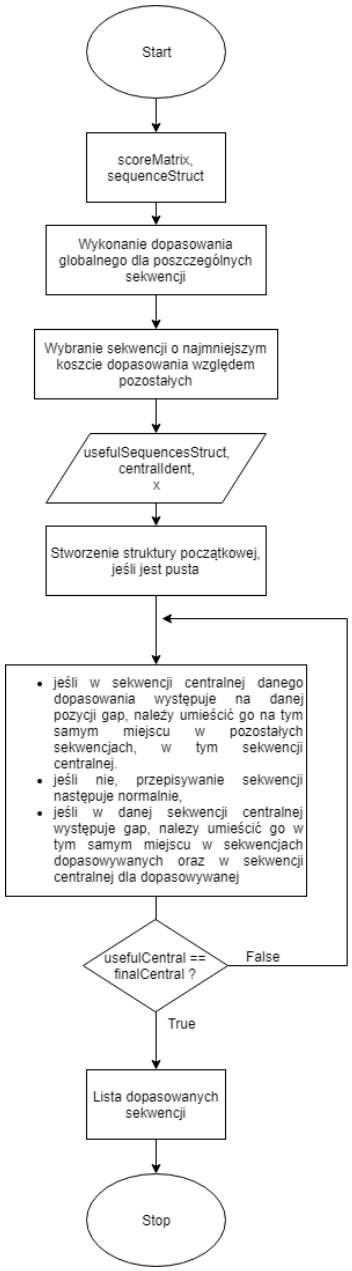
Dopasowanie dla gatunków z rodzaju 'Drosophila'.

```
>G8Drosophilahybri
TGG___T_TGA____A__A__A__CG____AA___________C_G__GCA_G___GCA__T____AA__
____C_____G_____T_____TTG_C___CT__T_T_T_GTTA_C__C_G___G_GC_GC_A___G_A_
_____A_______C__A___GA_AT_C__C_G_T__G__________AGA_GGA_A_A_AC_TCAG_G__
GA_A_A_AAG_AT__C____CT_CT__C_____CC

>T15Drosophilahybr
TGGACTT_TCA__TCA__A__A__TGTTTTTC___________C_C__GCATA___G_A__A____AACT
TTTTC_____G_____T___A_TTGTC_ATCT_AT_TCG_GTT_______G__TG_GCATC_A_TCGAA_
TTGTCA______CC__A___GA_AT____T_GTT__T__________ATA___ATC_A_AT_TC______
GATACT_TGGTAT_AC_TTGGTACT__C___AATA

>G11Drosophilahybr
T_G___TATAATT__AGCAGCAG_CG____AAGCTGCCGCCGCC_GCAGCA_GTT_GGAGGTGAACAA__
____C_____GAAACATACAGAATG_CGA_CT__TAT_TCATTT_C_TC_____A_AC_G__A_____AC
_____A__TTTT_C__A_TTTACTTCC_ACAGAT__G________CTACACGCATATC_GCGCCAG_GTT
TC_A_C_AA__A________CT_GTAGC_____CC

>C6Drosophilahybri
T_G___TATGA____A__AT_ATTC_____AT___________CTG__GAA_GATATCA__TG___AAT_
____CATTGAG_____T_____TCA_C___CTC_G_T_T_CCTACCA_CTGTT_GCGC__CTAC__A_A_
_____ATG_____CTTAT__AA_AT_CA_T_G_TAGGTCTTACAA__AGA_AGA_A_AGA____AGCA__
GA_A_AGAA__ATA_CA___CA_C___CAAG__GG
```

Plik .fasta

Działanie programu dla krótkich sekwencji z zajęć:

```
idB      __CGT___          1 - 8
idA      AACGT___          1 - 8
idC      __CGTAAA          1 - 8
idD      __CAT___          1 - 8
                             * *
```



Komentarze znajdujące się wewnątrz funkcji *multipleSequenceAligment.m* tłumaczą dokładnie działanie algorytmu.

multipleSequenceAligment.m

```
function [finalSequenceStruct] = multipleSequenceAligment(usefulSequencesStruct,x,centralIdent)
%MULTIPLESEQUENCEALIGMENT Summary of this function goes here
%   Detailed explanation goes here
finalSequenceStruct = struct;

%je?li dopasowywany powoduje gapa w centralnym -> dodaj gap do wszystkich
%poza dopasowywanym
%je?li mismatch to nic
%je?li centralny pierwszy ma gap, dodaj gap do dopasowywanego

%loop przez ilosc dopasowan jakie b?d? robione
for i = 1:x
    backup = usefulSequencesStruct;
    %jesli nie ma nic w finalnej strukturze
    if (isempty(fieldnames(finalSequenceStruct)))
        finalSequenceStruct.(centralIdent) = usefulSequencesStruct.("s" + i).(centralIdent);
        usefulSequencesStruct.("s" + i) = rmfield(usefulSequencesStruct.("s" + i),centralIdent);
        lastField = char(fieldnames(usefulSequencesStruct.("s" + i)));
        finalSequenceStruct.(lastField) = usefulSequencesStruct.("s" + i).(lastField);
    else
        %jesli struktury do dopasowania istnieja
        if(isfield(usefulSequencesStruct,char("s"+i)))
            %Centralna sekwencja z danego dopasowania
            usefulCentral = usefulSequencesStruct.("s" + i).(centralIdent);

            %usun z dopasowan zeby uzyskac pozosa??
            usefulSequencesStruct.("s" + i) = rmfield(usefulSequencesStruct.("s" + i),centralIdent);

            %Pozosta?a nazwa field sekwencji z danego dopasowania
            lastField = char(fieldnames(usefulSequencesStruct.("s" + i))); %dopasowywana

            %pozosta?a sekwencja
            lastSequence = char(usefulSequencesStruct.("s" + i).(lastField));

            %Field names w finalnej strukturze
            existingSequencesFields = fieldnames(finalSequenceStruct);


            c = 1;
            %porównujemy czy finalCentral == usefulCentral(którą będziemy
            %nadpisywac)
            while ~strcmp(usefulCentral,char(finalSequenceStruct.(char(existingSequencesFields(1)))))
                finalCentral = char(finalSequenceStruct.(char(existingSequencesFields(1))));
                %patrzymy czy jest gap w starym dopasowaniu
                if(numel(usefulCentral) >= c)
                    if(usefulCentral(c) == '_')
                        %sprawdzamy czy usefulCentral > finalCentral bo
                        %outofbound
                        if(numel(usefulCentral) <= numel(finalCentral))
                            %patrzymy czy jest gap w NOWYM dopasowaniu
                            if(finalCentral(c) == '_')
                                %nie rób nic jest gap jest w obu dopasowaniach
                            else
                                %jeśli w nowym dopasowaniu nie ma gapa
                                %nadpisz nowe dopasowanie + wszystkie
                                for n = 1:numel(existingSequencesFields)
                                    currentSeq = char(finalSequenceStruct.(char(existingSequencesFields(n))));
                                    currentField = char(existingSequencesFields(n));

                                    if(c ~= 1)
                                        currentSeq = char(strcat(currentSeq(1:c-1) + "_" + currentSeq(c:end)));
                                    elseif(c == numel(currentSeq))
                                        currentSeq = char(strcat(currentSeq + "_"));
                                    elseif(c == 1)
                                        currentSeq = char(strcat("_" + currentSeq));
                                    end
                                    finalSequenceStruct.(currentField) = currentSeq;
                                end
                            end
                        else
                            %finalCentral jest mniejsze of useful <-
                            %outofbound
                            for n = 1:numel(existingSequencesFields)
                                currentSeq = char(finalSequenceStruct.(char(existingSequencesFields(n))));
```

```
                    currentSeq = char(finalSequenceStruct.(char(existingSequencesFields(n))));
                    currentField = char(existingSequencesFields(n));
                    if(c ~= 1)
                        currentSeq = char(strcat(currentSeq(1:c-1) + "_" + currentSeq(c:end)));
                    elseif(c == numel(currentSeq))
                        currentSeq = char(strcat(currentSeq + "_"));
                    elseif(c == 1)
                        currentSeq = char(strcat("_" + currentSeq));
                    end
                    finalSequenceStruct.(currentField) = currentSeq;
                end
            end
            %jeśli finalna ma gapa, a w starym nie ma
        elseif(finalCentral(c) == '_')
            %dodaj gapy do dopasowywanej
            if(c ~= 1)
                lastSequence = char(strcat(lastSequence(1:c-1) + "_" + lastSequence(c:end)));
                usefulCentral = char(strcat(usefulCentral(1:c-1) + "_" + usefulCentral(c:end)));
            elseif(c == numel(lastSequence))
                lastSequence = char(strcat(lastSequence + "_"));
                usefulCentral = char(strcat(usefulCentral + "_"));
            elseif(c == 1)
                lastSequence = char(strcat("_" + lastSequence));
                usefulCentral = char(strcat("_" + usefulCentral));
            end
            %dopisz gap do nowej sekwencji
        elseif(numel(usefulCentral) < numel(finalCentral))
            lastSequence = char(strcat(lastSequence + "_"));
            usefulCentral = char(strcat(usefulCentral + "_"));
            %dodac warunek
        end
    end
    c = c + 1;
end
finalSequenceStruct.(lastField) = lastSequence;
end
end
end
usefulSequencesStruct = backup;
end
end
```

Złożoność czasowa:

n – ilość sekwencji wejściowych
k – górna granica sekwencji wejściowych (np. przy *for* w *while*)

- n – iteracji pierwszej pętli
  - maksymalnie k iteracji pętli while
    - maksymalnie n * k iteracji obu pętli for, więc *2nk*
    - maksymalnie 3k możliwych sprawdzeń warunku

$n * (k (2nk + 3k)) = 2k^2n^2 + 3k^2n = O(k^2n^2 + k^2n)$

Złożoność przestrzenna:

Maksymalny rozmiar sekwencji wejściowych: 2n * k <- n par sekwencji
Maksymalny rozmiar finalnej struktury: nk

2nk + nk = 3nk -> O(nk)