**K mean baseline**

Function recalcularCentre (Parameters: pointer to punts, N, D, C, pointer to

Centroides, pointer to PC, pointer to Sep)

{

Set i, j, k, x to zero

Set pointer v to malloc (size of D)

Set r to zero

Loop (Set i to zero; Check i less than to D; Increment i with one)

{

Loop (set x to zero; Check x less than to D; Increment x with one)

{

Set array v [index x] to zero

}

Loop (set j to zero; check j less than array Sep [index i]; Increment j with one)

{

Loop (set k to zero; check k less than to D; Increment k with one)

{

Set array v [index x] with sum of v [index x] and (Type casting

double) array Punts [ array PC [size is sum of

pointer to i, N, j] set sum of pointer D and k]

}

Loop (set x to zero; check x less than to D; Increment x with one)

{ set array v [index x] with division of array v [index x] and

(type casting with double) array Sep [index i]

If (array v [index x] not equal to Centroides)

set r to 1

set Centroides to array v [index x]

}

}

Free array v

Return (r)

}

Function generatePunts (parameters: pointer to Punts, N, D, C, pointer Centroides,

DIMENSION)

{

Set i, j to zero

Loop (set i to zero; check i less than to N; Increment i with one)

{

Loop (set j to zero; check j less than to D; Increment j with one)

{

Set Punts to remainder of function rand and DIMESNTION

}

}

Print (N and D)

Loop (set i to zero; check i less than to C; Increment i with one)

{

Set k to remainder of function rand and N

Loop (set j to zero; check j less than to D; Increment j with one)

{

Set Centroides to (type casting double) Punts

}

}

Print (new line and space)

}

Function mmin (parameters: pointer to v, C)

{

Set m to array v [index zero]

Set i to one and r to zero

Loop (set i to one; check i less than to C; Increment i with one)

{

If (m greater than array v [index i]

{

Set r to i and Set m to array v [index i]

}

}

Return (r)

}

Function PointsToCentroides (Parameter: pointer to Punts, N, D, C, pointer to

Centroides, pointer to PC, pointer to Sep)

{

Set i, j, k to zero

Set pointer dist to malloc (size of C)

Loop (set i to zero; check i less than to C; Increment i with one)

{

Set array Sep [index i] to zero

}

Loop (set i to zero; check i less than to N; Increment i with one)

{

Loop (set j to zero; check j less than to C; Increment j with one)

{

Set dist [index j] to zero

Loop (set k to zero; check k less than to D; Increment k with one)

{

Set array dist [index j] to (difference of Punts and Centroides for i)

multiplication of (Difference of punts and

Centroides for j)

}

}

Set m to Call function mmin (argument: dist, C)

Set array PC to i

Set array Sep to sum of array Sep and one

}

Free dist

}

Start

Function main (Parameters: argc, pointer to argv)

{

Declare N, D, C, DIMENSION

If (value of argc greater than five)

{

Set N to one thousand

Set D to three and Set C to four

Set DIMENSION to one thousand

}

Else {

Set N to call function atoi (Argument: argv [index one])

Set D to call function atoi (Argument: argv [index two])

Set C to call function atoi (Argument: argv [index three])

Set DIMENSION to call function atoi (Argument: argv [index four])

}

Set pointer Punts to malloc (size of pointer N and D)

Set pointer Centroides to malloc (size of pointer C and D)

Set pointer PC to malloc (size of pointer N and C)

Set pointer Sep to malloc (size of pointer C)

Set cont and final to zero

Print (N, D, C)

Print (DIMENSION, D)

Call function generatePunts (Arguments: punts, N, D, C, Centroides,

DIMENSION)

Do

{

Call function PointsToCentroides (Arguments: Punts, N, D, C, Centroides,

Sep)

Set cont to sum of cont and one

Set final to call function recalcularCentre (Arguments: Punts, N, D, C,

Centroides, PC, Sep)

}

While (final or cont less than to ten thousand)

{

Print (cont)

Set i and j to zero

Loop (set i to zero; check i less than C; Increment i with one)

{

Loop (set j to zero; check j less than D; Increment j with one)

{

Print (Centroides)

}

}

}

}