\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: hw7

Author: F74046022 陳冠仁 <jeremy851004@gmail.com>

Class: 乙班

Description:

載入buckets.in並傳換成陣列bi[][]

以bi[][]得出要輸出至buckets.out的陣列bo[][]

以bi[][]、bo[][]得出要輸出至mapping.out的陣列map[][]

分別輸出bo[][]、map[][]至buckets.out、mapping.out

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include <stdio.h>

int i, j, x, y;

//loop variables

char ch;

//temporary char

int main(int argc, char \*argv[])

{

FILE \*bc\_in, \*bc\_out, \*map\_out;

bc\_in=fopen("/home/data/hw7/buckets.in", "r");

bc\_out=fopen("buckets.out", "w");

map\_out=fopen("mapping.out", "w");

if(!bc\_in)

{

printf("ERROR\n");

return 0;

}

int line, size;

line\_size(bc\_in, &line, &size);

//line -> number of lines in buckets.in

//size -> maximum number of numbers in a line

int n=input\_n(size);

if(n<size)

{

printf("ERROR\n");

return 0;

}

//input n (>=size)

int bi[line][n], bo[line][n], map[line][n];

int elm[line], idx[line], idx\_max;

//bi[][] -> array for buckets.in

//bo[][] -> array for buckets.out

//map[][] -> array for mapping.out

//elm[] -> record how many numbers in each line

//idx[] -> record the bucket index of each line

//idx\_max -> the maximum number of bucket used

load\_bi(bc\_in, line, n, elm, bi);

//transform buckets.in into bi[][]

fill\_bo(line, n, &idx\_max, elm, bi, bo);

//use bi[][] to generate bo[][]

gnr\_map(line, n, elm, bi, bo, map, idx);

//use bi[][], bo[][] to generate map[][], idx[]

output\_bo(bc\_out, n, idx\_max, bo);

output\_map(map\_out, line, n, map, idx);

printf("TOTAL NUMBER OF BUCKETS : %d\n\n", idx\_max);

fclose(bc\_in);

fclose(bc\_out);

fclose(map\_out);

return 0;

}

int line\_size(FILE \*f, int \*line, int \*size)

{

\*line=1;

\*size=0;

int cnt=1;

while(fscanf(f, "%c", &ch)!=EOF)

{

if(ch==' ') cnt++;

if(ch=='\n')

{

\*line=\*line+1;

if(cnt>\*size) \*size=cnt;

cnt=1;

}

}

if(cnt>\*size) \*size=cnt;

return;

}

int input\_n(int size)

{

int a;

printf("\nPLEASE ENTER N (>=%d) : ", size);

scanf("%d", &a);

return a;

}

int load\_bi(FILE \*f, int line, int n, int elm[], int bi[][n])

{

for(i=0; i<line; i++)

elm[i]=1;

//initialize elm[]

i=0;

rewind(f);

while(fscanf(f, "%c", &ch)!=EOF)

{

if(ch==' ') elm[i]++;

if(ch=='\n') i++;

}

//count how many numbers in each line

rewind(f);

for(i=0; i<line; i++)

for(j=0; j<elm[i]; j++)

fscanf(f, "%d", &bi[i][j]);

//load buckets.in into bi[][]

return;

}

int fill\_bo(int line, int n, int \*idx\_max, int elm[], int bi[][n], int bo[][n])

{

\*idx\_max=0;

for(i=0; i<line; i++)

for(j=0; j<n; j++)

bo[i][j]=0;

//initialize array for buckets.out

int bi\_rem[line], bo\_rem[line];

for(i=0; i<line; i++)

{

bi\_rem[i]=elm[i];

bo\_rem[i]=n;

}

//bi\_rem[] -> how many number needs to be put in

//bo\_rem[] -> the remaining spaces of buckets

for(i=0; i<line; i++)

{

for(x=0; x<line; x++)

{

for(j=0; j<elm[i]; j++)

for(y=0; y<n; y++)

if(bi[i][j]==bo[x][y])

{

bi\_rem[i]--;

break;

}

if(bi\_rem[i]>bo\_rem[x])

bi\_rem[i]=elm[i];

else

break;

}

//check if bo[x][] can be used

for(j=0; j<elm[i]; j++)

for(y=0; y<n; y++)

{

if(bi[i][j]==bo[x][y])

break;

if(bo[x][y]==0)

{

bo[x][y]=bi[i][j];

bo\_rem[x]--;

break;

}

}

//fill bo[x][] with bi[i][]

if(x+1>\*idx\_max) \*idx\_max=x+1;

//record the maximum index used

}

return;

}

int gnr\_map(int line, int n, int elm[], int bi[][n], int bo[][n], int map[][n], int idx[line])

{

for(i=0; i<line; i++)

for(j=0; j<n; j++)

map[i][j]=0;

//initialize the array for mapping.out

int temp;

//variable for checking

for(i=0; i<line; i++)

{

for(x=0; x<line; x++)

{

temp=0;

for(j=0; j<elm[i]; j++)

for(y=0; y<n; y++)

if(bi[i][j]==bo[x][y])

{

temp++;

break;

}

if(temp==elm[i])

break;

}

//check if bo[x][] can be used

idx[i]=x;

//mark the index of bucket

for(j=0; j<elm[i]; j++)

for(y=0; y<n; y++)

if(bi[i][j]==bo[x][y])

{

map[i][y]=1;

break;

}

//generate map[i][] according to bi[i][], bo[x][]

}

return;

}

int output\_bo(FILE \*bc\_out, int n, int idx\_max, int bo[][n])

{

for(i=0; i<idx\_max; i++)

{

for(j=0; j<n; j++)

fprintf(bc\_out, "%d ", bo[i][j]);

fprintf(bc\_out, "\n");

}

}

int output\_map(FILE \*map\_out, int line, int n, int map[][n], int idx[])

{

for(i=0; i<line; i++)

{

fprintf(map\_out, "%d ", idx[i]);

for(j=0; j<n; j++)

fprintf(map\_out, "%d", map[i][j]);

fprintf(map\_out, "\n");

}

}

Compilation:

gcc –o hw7 hw7.c

Execution:

./hw7

Output:

F74046022@c-2015-2:~/hw7> gcc -o hw7 hw7.c

F74046022@c-2015-2:~/hw7> ./hw7

PLEASE ENTER N (>=38) : 45

TOTAL NUMBER OF BUCKETS : 13

F74046022@c-2015-2:~/hw7> ./hw7\_checker

no error

hw7.c: In function ‘line\_size’:

hw7.c:66:2: error: incompatible type for argument 1 of ‘fscanf’

while(fscanf(f, "%c", &ch)!=EOF)

^

In file included from /usr/include/features.h:364:0,

from /usr/include/stdio.h:27,

from hw7.c:1:

/usr/include/stdio.h:443:12: note: expected ‘struct FILE \* \_\_restrict\_\_’ but argument is of type ‘FILE’

extern int \_\_REDIRECT (fscanf, (FILE \*\_\_restrict \_\_stream,

^

hw7.c:87:1: error: expected ‘;’ before ‘}’ token

}

^

hw7.c:232:1: error: expected declaration or statement at end of input

}

^

hw7.c: In function ‘load\_bi’:

hw7.c:96:2: error: incompatible type for argument 1 of ‘rewind’

rewind(\*f);

^

In file included from hw7.c:1:0:

/usr/include/stdio.h:759:13: note: expected ‘struct FILE \*’ but argument is of type ‘FILE’

extern void rewind (FILE \*\_\_stream);

^