201533661 이승수’s Algorithm homework#2 Date:2016.09.07

<code>

#include <stdio.h>

FILE \*inF, \*outF;

int tempPlate[30] = { 0 };

int tempFront = 0, tempRear = 0;

void push(int value);

int pop();

int Flip(int Plate[], int index, int Size);

int findMaxIndex(int Plate[], int indexStart, int Size);

int findMaxValue(int Plate[], int indexStart, int Size);

int Descending(int Plate[], int indexStart, int Size);

void main()

{

int number=0;

int MainPlate[30] = { 0 };

int Index = 1;

int FlipQueue[61] = { 0 },flipRear=0;

inF = fopen("input#1.txt", "r");

while (!feof(inF))

{

fscanf(inF,"%d",&MainPlate[Index++]);

number++;

}

fclose(inF);

outF = fopen("output#1.txt","w");

for (int i = 1; i <=number; i++)

{

fprintf(outF," %d",MainPlate[i]);

}

Flip(MainPlate, 1, number);/\*just flip for using index in ascending order\*/

//flipping

for (int i = 1; i <= number; i++)

{

int MaxIndex = findMaxIndex(MainPlate, i, number);

if (Descending(MainPlate, i, number))//if already sorted.

break;

else if (MainPlate[i] == MainPlate[MaxIndex])//if Max pancake is at the range's bottom

continue;

else

;

int TempforFlip = Flip(MainPlate, MaxIndex, number);

if (TempforFlip&&(TempforFlip!=i))

{

if (TempforFlip!=number)

FlipQueue[flipRear++] = TempforFlip;

Flip(MainPlate,i,number);

FlipQueue[flipRear++] = i;

}

else if (TempforFlip&&(TempforFlip==i))

{

Flip(MainPlate,i,number);

}

else//TempforFlip==0

{

;

}

}

//(printing ordered)

Flip(MainPlate,1,number);/\*Flip again because flip at ahead for ascending index\*/

fprintf(outF,"(");

for (int i = 1; i <= number; i++)

{

fprintf(outF,"%d ",MainPlate[i]);

}

fprintf(outF,")\n");

//\nprint flip queue

for (int i = 0; i <= flipRear; i++)

{

fprintf(outF,"%d ",FlipQueue[i]);

}

fclose(outF);

}//end main

void push(int value)

{

tempPlate[tempRear++] = value;

}

int pop()

{

int storage;

storage = tempPlate[--tempRear];

return storage;

}

int Flip(int Plate[],int index,int Size)

{

if (index >= 1)

;

else

return 0;

for (int i = index; i <= Size; i++)

{

push(Plate[i]);

}

for (int i = index; i <= Size; i++)

{

Plate[i]=pop();

}

return index;

}

int findMaxIndex(int Plate[],int indexStart,int Size)

{

int Index=indexStart;

for (int i = indexStart; i <= Size; i++)

{

if (Plate[Index] <= Plate[i])

Index = i;

}

if (Index == indexStart)

return 0;

else

return Index;

}

int findMaxValue(int Plate[],int indexStart,int Size)

{

int Max = 0;

Max = Plate[findMaxIndex(Plate,indexStart,Size)];

return Max;

}

int Descending(int Plate[], int indexStart,int Size)

{

for (int i = indexStart; i < Size;i++)

{

if (Plate[i] < Plate[i + 1])

return 0;

}

return 1;//this plate is descending order

}