**ALGORITHM**

201133216

정유석

EXERCISE 2 - **Delicious Pancakes**

#include<stdio.h>

#include<conio.h>

#include <stdio.h>

#include<stdlib.h>

int list[30] ,top=0;

int push(int \*, int, int);

int pop(int \*, int);

int full();

int empty();

int p\_sort(int \*, int);

void flip(int\*, int, int);

int main() {

int i,n,cnt;

printf("Enter the number of input (1~30) : ");// Each stack will consist of between 1 and 30 pancakes

scanf("%d",&n);

where( n>30 || n<1){

printf(“Re Input number of elephant : “);

scanf(“%d”,&n);

}

top = n;

top = push(list, n, top); //Receive the stack

cnt = p\_sort(list,n); //Pancakes sort

top = pop(list,top); //Pop the stack

printf("\nSwap count : %d \n", cnt);

return 0;

}

void flip(int \*list, int length, int num){ //Flip function

int temp; // temporary value

int i=0; // loop value

for(i;i<--num;i++){ //Flip

temp=list[i];

list[i]=list[num];

list[num]=temp;

}

}

int p\_sort(int \*list, int length) { //Pancakes sort

int i,a,max\_n\_position,swap;

swap=0;

//If it's less than 2 long, just return

for(i=length;i>1;i--) {

//Find the max number position

max\_n\_position=0;

for(a=0;a<i;a++) {

if(list[a]>list[max\_n\_position])

max\_n\_position=a;

}

if(max\_n\_position==i-1) //No flip -- if the max number is down floor

continue;

//find max number to the beginning of the list

if(max\_n\_position){

swap++;

flip(list, length, max\_n\_position+1);

}

swap++;

flip(list, length, i);

//Then list[i-1] is sorted

}

return swap;

}

int push(int \*list, int n, int top){

printf("\n\nEnter the diameters (1~10) :\n"); //each pancake will have an integer diameter between 1 and 10.

for ( top = 0; top < n; top++) {

if(!full){ //Check the stack

printf("Wrong stack ! \n");

exit(0);

}

scanf("%d",&list[top]);

if(list[top]>10 && list[top]<1) //Exception

exit(0);

}

return top;

}

int pop(int \*list, int top){ //pop function

int i;

printf("\n\nComplete pancakes!\n");

top--;

while(top>=0){

printf("%d ",list[top]);

list[top] = 0; //initialize the stack

top--;

if(!empty){ //Check the stack

printf("Stack is Empty ! \n");

break;

}

}

return top;

}

int full(){ //Check the stack full

return top==30;

}

int empty(){ // Check the stack empty

return top==0;

}



