# Algoritma Analizi 1.Ödev Raporu

# Ömer Lütfü Tortumlu

# 16011110

# YÖNTEM

-Kullanıcıdan alınan n değeriyle calloc() fonksiyonuyla bir pointer dizi tanımlanır.

-Kullanıcıdan çekirgenin hangi taştan atlamaya başlaması gerektiği bilgisi alınır.(Kullanıcı diziyi kapsayacak bir değer girmeli.)

-Dizinin(taşın üstündeki sayılar) değerleri kullanıcıdan son indis hariç alınır.(Son indis calloc() fonksiyonuyla 0 atanıyor.)

-Dizi(leapCount),n(Dizi boyutu) ve başlangıç noktası(Start) recursive Solvable() fonksiyonuna gönderilir.

-Fonksiyon gelen başlangıç noktasının dizinin sınırları içinde olup olmadığını her çağrıldığında kontrol eder.

-Başlangıç noktasının son indise(n-1) eşit olmadığı kontrolü yapılır.

-Başlangıç noktasıyla başlangıç değerinin toplamı n den büyükse sol tarafa sıçrama yapılır.

-Eğer sol tarafa sıçramada değer 0’dan küçükse oyun başarısız olarak sonuçlanır.

( Örneğin: 1-1-5-1-1) başlangıç 3 olarak ayarlanırsa sağdan ve soldan sıçrama yapılamaz.

-Arttırılan başlangıç değeri tmp1 azaltılan başlangıç değeri de tmp2 değişkeninde tutulur. Bu değişkenler birbirine eşit olduğu ve sağa sola gidişte dizi dışında kaldığı zaman sonsuz döngüye gireceği için oyun başarısız olarak sonuçlanır.

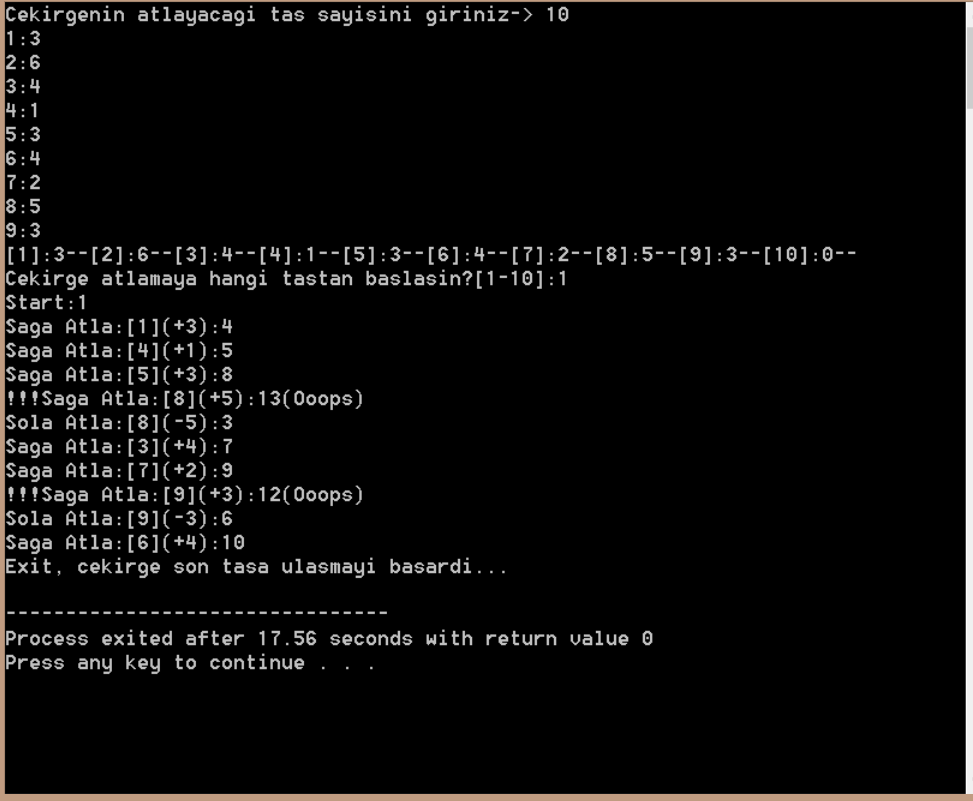
-Arttırılan başlangıç değeri tmp1 sınırı aşıyorsa ve azaltılan değer(sola kaydırılan tmp2) değer tmp1 ile aynıysa başlangıç noktası 2 kere sola kaydırılır, aynı değilse 1 kere sola kaydırılır.

-Başlangıç ile başlangıç indisinin değerinin toplamı (leapCount[start]+start<n) n den küçükse başlangıç değeri 1 kere sağa kaydırılır ve sağa kaydırılma değeri tmp2 de tutulur.

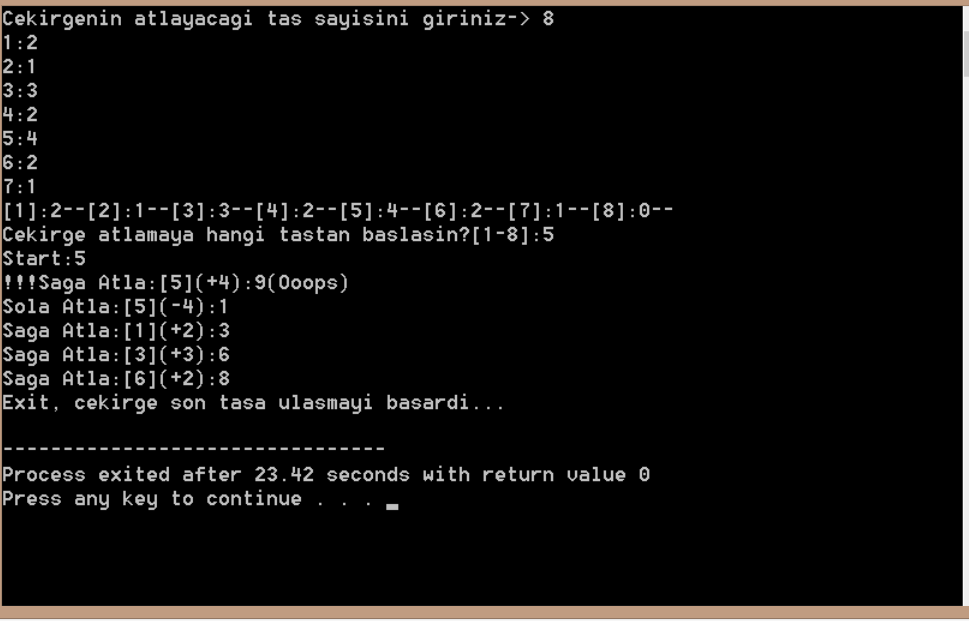
-En son indise ulaşılabiliyorsa fonksiyon 1 ulaşılamazsa 0 değerini dönderir.

# UYGULAMA

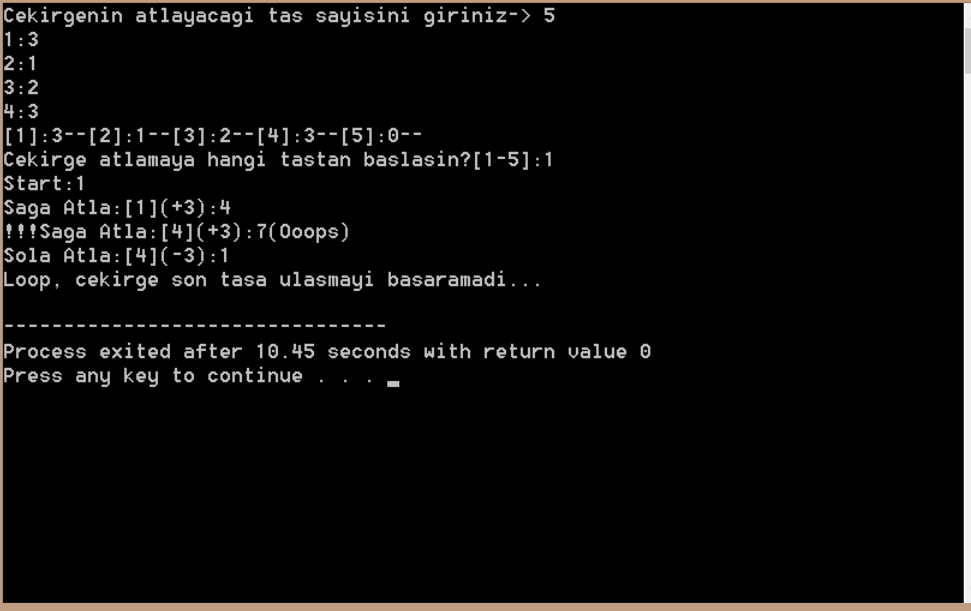
1.Son taşa ulaşan;



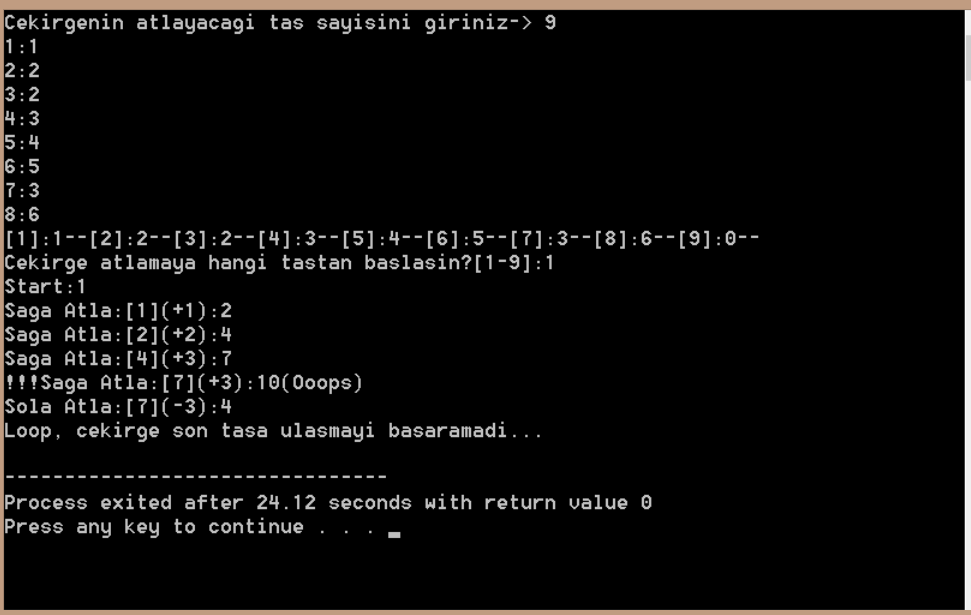
2.Son taşa ulaşan;



1.Son taşa ulaşamayan;



2.Son taşa ulaşamayan;



# KOD

#include<stdio.h>

#include <stdlib.h>

#include <conio.h>

int donus; //solvable fonksiyonundan dönüş değeri

int tmp1=1000; //Arttırılan başlangıç değeri tmp1

int tmp2=1000; //Azaltılan başlangıç değeri de tmp2

int solvable(int start,int leapCount[],int n){

if(start<0 || start>=n){ //Start değişkeninin belirlediğimiz dizi içinde olduğunu kontrol ediyor.

return 0;

}

if(start==n-1){ //Start değişkeninin son indise ulaştığını kontrol ediyor.

return 1;

}

if(leapCount[start]+start>=n){ //start ile indisin değerinin toplamı n den buyuk olursa bir sola geliyor

printf("!!!Saga Atla:[%d](+%d):%d(Ooops)\n",start+1,leapCount[start],leapCount[start]+start+1);

printf("Sola Atla:[%d](-%d):%d\n",start+1,leapCount[start],start-leapCount[start]+1);

start=start-leapCount[start];

if(start<0){ //( Örneğin: 1-1-5-1-1) başlangıç 3 olarak ayarlanırsa sağdan ve soldan sıçrama yapılamaz.

return 0;

}

tmp1=leapCount[start]; //Arttırılan başlangıç değeri tmp1, azaltılan başlangıç değeri de tmp2 değişkeninde tutulur.

if(tmp1==tmp2 ){ //Eğer arttırımda sınır aşılıyorsa ve tmp ler eşit çıkarsa bir sola daha kayılır.

if(start+tmp1>=n && start-tmp2<0)

return 0;

else if(start-tmp2>0){ //tmp1 ile tmp2 aynı değilse iki kere sola gidilir

printf("Sola Atla:[%d](-%d):%d\n",start+1,leapCount[start],start-leapCount[start]+1);

start=start-leapCount[start];

donus=solvable(start,leapCount,n);

if(donus==1) return 1;

if(donus==0) return 0;

}

}

else{

donus=solvable(start,leapCount,n); //tmp1 ile tmp2 aynı değilse bir kere sola gidilir

if(donus==1) return 1;

if(donus==0) return 0;

}

}

else{

printf("Saga Atla:[%d](+%d):%d\n",start+1,leapCount[start],leapCount[start]+start+1);

start=leapCount[start]+start; //start ile indisin değerinin toplamı n den kucukse

tmp2=leapCount[start]; //Azaltılan başlangıç değeri tmp2 değişkeninde tutulur.

donus=solvable(start,leapCount,n); //starta yeni değeri ataniyor

if(donus==1) return 1;

if(donus==0) return 0;

}

}

int main(){

// Dinamik bir dizi yaratmak için pointer kullanılır.

int n,i,start,\*leapCount; // Dizimizin kac elemanli olacagini n de, cekirgenin nerden başlıyacağını startta tutuyorum.i yi indis işlemlerinde kullanıyorum

printf( "Cekirgenin atlayacagi tas sayisini giriniz-> "); // Kullanicidan tas sayisini girmesini istiyoruz.

scanf( "%d", &n ); //Toplam taş sayısı girilir

leapCount= calloc( n, sizeof( int ) ); // calloc( ) fonksiyonuyla dinamik olarak dizi girilen boyuta göre ayarlanır.

for(i = 0; i < n-1; i++ ){ //Taşların üstündeki sayı değerlerini alıyoruz son taş haric(0)

printf( "%d:", i+1 );

scanf("%d",&leapCount[i]);

}

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

printf( "[%d]:%d--", i+1,leapCount[i] ); //Taşların üstündeki sayı değerlerini yazdırıyoruz.

}

printf("\nCekirge atlamaya hangi tastan baslasin?[1-%d]:",n); //Start değişkenini kullanıcıdan istiyoruz

scanf("%d",&start); //Kullanıcı tarafından start değişkeni giriliyor.

printf("Start:%d\n", start,leapCount[start]); //Başlanılacak taştaki bilgi veriliyor

donus=solvable(start-1,leapCount,n); //Cekirgenin taşlardan çıkıp çıkmadığı bilgisi recursive fonksiyon yardımıyla hesaplanıyor

if(donus==1){ //Cekirgenin son taşa ulaşıp ulaşamadığı yazdırılır.

printf("Exit, cekirge son tasa ulasmayi basardi...\n");

}

else

printf("Loop, cekirge son tasa ulasmayi basaramadi...\n");

free(leapCount);// Dinamik olan diziyi kullandiktan sonra free fonksiyonunu kullanıyorum.

return 0;

//Ömer Lütfü Tortumlu 16011110

}