מגישים : שמעון דסטה ת.ז 203670286

טל סער ת.ז 209151380

פתרונות תרגיל מספר 3:

שאלה מספר 1:

מצ"ב קובץ zip הכולל פתרון מלא עם פונקציית main .

ובנוסף מצורף הקוד :

#pragma once

typedef int queueinfo;

typedef struct node

{

queueinfo value;

struct node\* next;

} QueueNode;

QueueNode\* createQueue();

int isEmptyQueue(QueueNode\* Q);

void insert(QueueNode\*\* Q, queueinfo x);

void delafter(QueueNode\* p, queueinfo\* x);

void printQueue(QueueNode\* Q);

void insertList(int number, int sum);

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "queue.h"

QueueNode\* createQueue()

{

return NULL;

}

int isEmptyQueue(QueueNode\* Q)

{

if (Q == NULL)

return ;

}

void insert(QueueNode\*\* Q, queueinfo x)

{

QueueNode\* node = (QueueNode\*)malloc(sizeof(QueueNode));

if (!(\*Q))

{

(\*Q) = node;

node->value = x;

node->next = node;

}

else

{

node->value = x;

node->next = (\*Q)->next;

(\*Q)->next = node;

}

}

void delafter(QueueNode\* p, queueinfo\* x)

{

QueueNode\* node = p->next->next;

\*x = p->next->value;

free(p->next);

p->next = node;

}

void printQueue(QueueNode\* Q) {

if (isEmptyQueue(Q) == 0)

printf("%d ", Q->value);

}

void insertList(int num, int sum)

{

QueueNode\* node = NULL;

queueinfo value = 0;

int i;

for ( i = 1; i <= num; i++)

{

insert(&node, (num - i )+ 1);

}

printf("Order of elimination: ");

while (node->next != node)

{

for ( i = 0; i < sum - 1; i++)

{

node = node->next;

}

delafter(node, &value);

printf("%d ", value);

}

delafter(node, &value);

printf("\nLeader: %d", value);

}

int main()

{

int numSoldiers = 0;

int skip = 0;

printf("\n");

printf("Please enter the number of soldiers : \n");

scanf("%d", &numSoldiers);

printf("Please enter the skip number :\n");

scanf("%d", &skip);

insertList(numSoldiers, skip);

}

שאלה מספר 2:

מצ"ב קובץ zip הכולל פתרון מלא עם פונקציית main .

ובנוסף מצורף הקוד :

#pragma once

typedef struct node

{

int key;

struct node\* next;

}NODE;

NODE\* createList();

int isEmptyList(NODE\* head);

NODE\* ReorderFirst(NODE\* head);

void ReorderSecond(NODE\*\* head);

void insert(NODE\*\* n, int k);

void printList(NODE\* n);

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Node.h"

NODE\* createList()

{

return NULL;

}

int isEmptyList(NODE\* head)

{

return head == NULL;

}

NODE\* ReorderFirst(NODE\* head)

{

NODE\* even = NULL,\* odd = NULL;

NODE\* headEven = NULL;

NODE\* headOdd = NULL;

while (head) {

if (head->key % 2) {

if (odd)

odd->next = head;

else

headOdd = head;

odd = head;

}

else {

if (even) even->next = head;

else headEven = head;

even = head;

}

head = head->next;

}

if (!odd) return headEven;

if (!even) return headOdd;

odd->next = headEven;

even->next = NULL;

return headOdd;

}

//part 2

void ReorderSecond(NODE\*\* head)

{

NODE\* temp = \*head,\* start;

NODE\* odd = NULL,\*sumOdd = NULL;

NODE\* even = NULL,\*sumEven = NULL;

int counter;

while (temp != NULL)

{

counter = temp->key;

start = temp;

temp = temp->next;

if (counter % 2 == 0)

{

if (sumEven == NULL)

{

sumEven = start;

even = sumEven;

}

else

{

sumEven->next = start;

sumEven = sumEven->next;

}

}

else

{

if (sumOdd == NULL)

{

sumOdd = start;

odd = sumOdd;

}

else

{

sumOdd->next = start;

sumOdd = sumOdd->next;

}

}

}

if (sumOdd != NULL)

{

sumOdd->next = even;

\*head = odd;

if (sumEven != NULL)

{

sumEven->next = NULL;

}

}

else

{

\*head = even;

}

}

void insert(NODE\*\* n, int k)

{

if (\*n == NULL)

{

\*n = (NODE\*)malloc(sizeof(NODE));

(\*n)->next = NULL;

(\*n)->key = k;

}

else

{

NODE\* node = \*n;

while (node->next != NULL)

{

node = node->next;

}

node->next = (NODE\*)malloc(sizeof(NODE));

node->next->key = k;

node->next->next = NULL;

}

}

void printList(NODE\* n)

{

while (n->next != NULL)

{

printf("%d->", n->key);

n = n->next;

}

printf("%d\n", n->key);

}

int main()

{

int i, reorderList;

int num = 0;

NODE\* listNum = NULL;

NODE\* listNumtwo = NULL;

printf("Please enter a number for the list:\n");

scanf("%d", &num);

printf("Enter a numbers to add to list:\n");

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

{

scanf("%d", &reorderList);

insert(&listNum, reorderList);

insert(&listNumtwo, reorderList);

}

printf("\n");

printList(listNum);

printf("\n");

NODE\* newList = ReorderFirst(listNum);

printf("The new linked list \n");

printList(newList);

printf("\n");

ReorderSecond(&listNumtwo);

printf("linked list in one passage\n");

printList(listNumtwo);

free(listNumtwo);

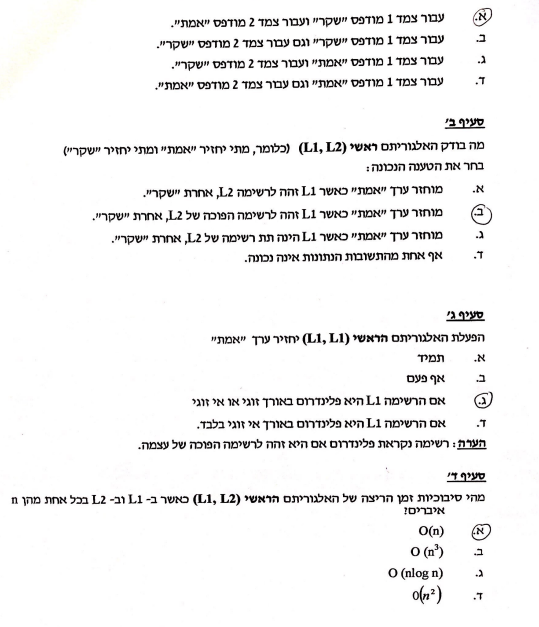
return 0;

}

שאלה מספר 3:

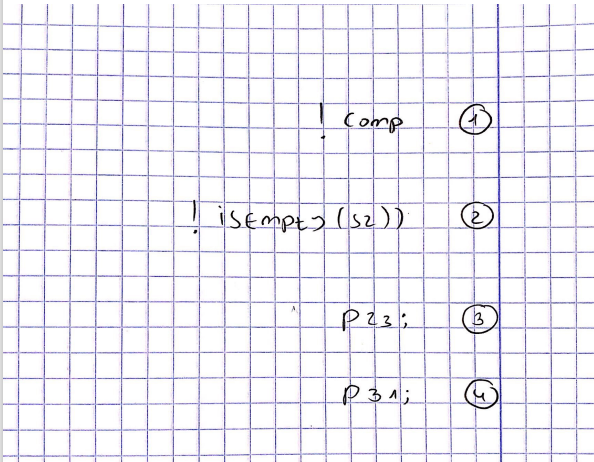
|  |  |
| --- | --- |
| סעיף | פתרון |
| א | א |
| ב | ב |
| ג | ג |
| ד | א |

מצ"ב צילום פתרון:

** סעיף א'**

שאלה מספר 4:

מצ"ב השלמה לפי מספר 1-4:



ומצ"ב פתרון קוד:

while (!(IsEmpty(S1)))

{

P12;

while (!(IsEmpty(S1)))

{

if (!Comp)

{

P23;

P12;

P31;

}

P12;

}

P23;

while (!isEmpty(S2))

{

P23;

P31;

}

}

Print(S3);

שאלה מספר 5:

מצ"ב קובץ zip הכולל פתרון מלא עם פונקציית main .

#pragma once

typedef itemType;

typedef struct binTree {

int info;

struct binTree\* right;

struct binTree\* left;

}treeNode, \* tree\_ptr;

void rangeSearch(tree\_ptr t, itemType u, itemType v);

tree\_ptr Binary(tree\_ptr n1, tree\_ptr n2, int sum);

void insert(tree\_ptr\* ptr, int val);

void printTree(tree\_ptr ptr);

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Tree.h"

void rangeSearch(tree\_ptr t, itemType u, itemType v)

{

if (t == NULL)

return;

rangeSearch(t->left, u, v);

if (t->info >= u && t->info <= v)

{

printf("%2d", t->info);

}

rangeSearch(t->right, u, v);

}

tree\_ptr Binary(tree\_ptr n1, tree\_ptr n2, int sum)

{

tree\_ptr type = (tree\_ptr)malloc(sizeof(treeNode));

if (!type)

return NULL;

type->info = sum;

type->left = n1;

type->right = n2;

return type;

}

void insert(tree\_ptr\* ptr, int val)

{

if (val == (\*ptr)->info)

return;

if (val < (\*ptr)->info)

{

if (NULL == ((\*ptr)->left))

(\*ptr)->left = Binary(NULL, NULL, val);

else

insert(&(\*ptr)->left, val);

}

else

{

if (NULL == ((\*ptr)->right))

(\*ptr)->right = Binary(NULL, NULL, val);

else

insert(&(\*ptr)->right, val);

}

}

void printTree(tree\_ptr ptr)

{

if (!ptr)

return;

printf("%d ", ptr->info);

printTree(ptr->left);

printTree(ptr->right);

}

int main() {

int u = 4, v = 9;

tree\_ptr ptr = (tree\_ptr)malloc(sizeof(treeNode));

ptr = Binary(NULL, NULL, 0);

for (int i = 1; i < 15; i++)

insert(&ptr, i);

printf("Printing Tree:\n");

printTree(ptr);

printf("\n");

printf("The Range Search(u=%d v=%d):\n", u, v);

rangeSearch(ptr, u, v);

printf("\n");

free(ptr);

return 0;

}

שאלה מספר 6:

מצ"ב פתרון קוד(השלמת ביטויים -המודגשים ) :

סעיף א

**אלגוריתם**

**צעד :1**  נסרוק את העץ ב **inorder** ותוך כדי הסריקה להעתיק את אברי העץ למערך שגדלו n

כגודל העץ. )כמובן נרצה שהמערך יהיה ממוין(.

**צעד 2:** נזמן את השגרה הרקורסיבית שלהלן )מופיעה מטה( ע"פ הזימון הבא:

tree\_ptr p = NULL;

Trouble (0, n-1, &p);

הפונקציה הרקורסיבית Trouble שמטה מקבלת שני אינדקסים i ו - j במערך ממוין ובונה

באמצעות אברי המערך A[i]….A[j] עץ חיפוש בינארי **מאוזן במספר צמתיו** שלשורשו יצביע t .

void Trouble (int i, int j, tree\_ptr \*t)

{

int mid ;

if (i <= j) {

if (i = = j) {

\*t = (tree\_ptr\*) malloc (sizeof(tree\_ptr));

(\*t) ->info =A[i];

**return**;

}else {

mid= **(i + j) / 2**;

\*t = (tree\_ptr\*) malloc (sizeof(tree\_ptr));

(\*t) ->info =A[mid];

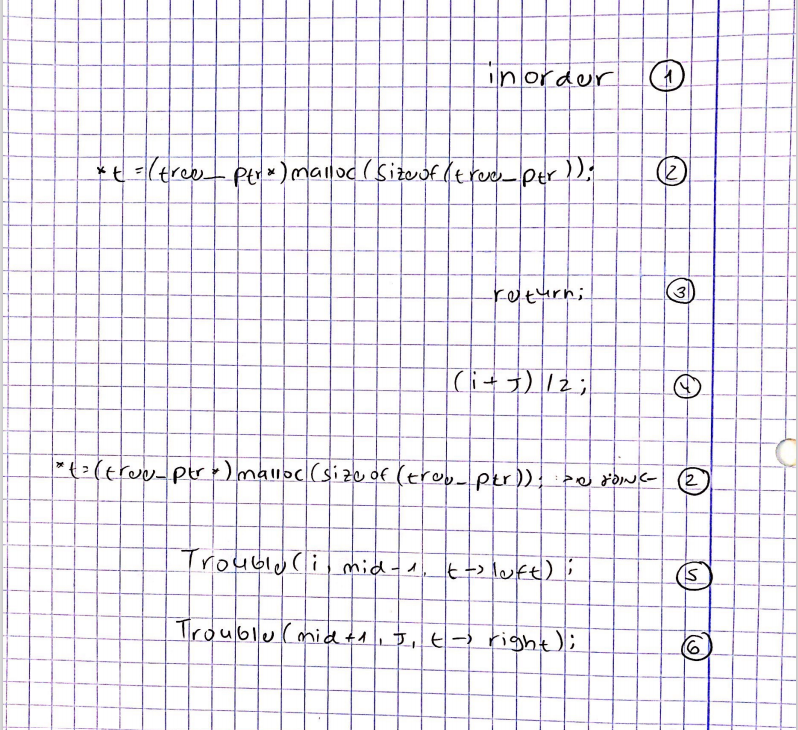
**Trouble(i, mid – 1, t->left)**;

**Trouble(mid + 1, j, t->right)**;

}

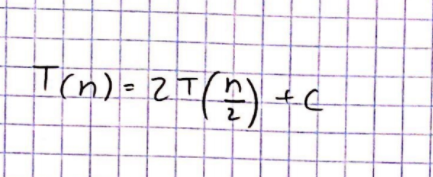
}

}

ובנוסף מצ"ב השלמת ביטויים 1-6:

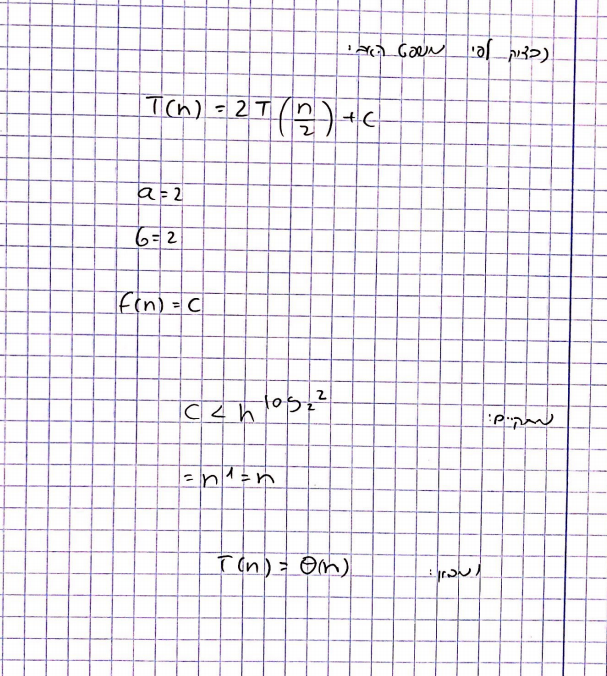
סעיף ב':

פונקציה רקורסיבית בעבור T(n)

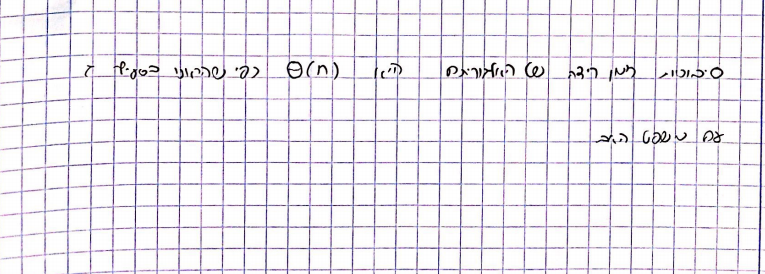


סעיף ג' :

נבדוק סיבוכיות זמן ריצה :



סעיף ד' :

נקבע סיבוכיות זמן ריצה של האלגוריתם :

שאלת בונוס :

מצ"ב קובץ zip עם פתרון לשאלת בונוס .