**תרגיל בית מספר 11 – מבנים**

איתי חסיד  
316166636

שאלה 1

Real struct2array(**int** num, **int** frac)

{

Real newArr;

**int** countOfDigitsNum = 0, countOfDigitsFrac = 0, tempNum = num, tempFrac = frac;

**while** (num != 0)

{

num = num / 10;

countOfDigitsNum++;

}

**while** (frac != 0)

{

frac = frac / 10;

countOfDigitsFrac++;

}

newArr.num = (**char**\*)malloc((countOfDigitsNum + 1) \* **sizeof**(**char**));

newArr.frac = (**char**\*)malloc((countOfDigitsFrac + 1) \* **sizeof**(**char**));

**if** (newArr.num == **NULL** || newArr.frac == **NULL**)

{

**return** newArr;

}

**else**

{

**for** (**char**\* p = newArr.num + countOfDigitsNum - 1; p >= newArr.num; p--)

{

\*p = (**char**)(tempNum % 10+48);

tempNum = tempNum / 10;

}

newArr.num[countOfDigitsNum] = 0;

**for** (**char**\* p = newArr.frac + countOfDigitsFrac - 1; p >= newArr.frac; p--)

{

\*p = (**char**)(tempFrac % 10+48);

tempFrac = tempFrac / 10;

}

newArr.frac[countOfDigitsFrac] = 0;

**return** newArr;

}  
}

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שאלה 2

**char** \*struct2array1(Real \*s)

{

**long** **long** count1 = strlen(s->num);

**long** **long** count2 = strlen(s->frac);

**char** \*newString = (**char** \*)malloc((count1 + count2 + 2) \* **sizeof**(**char**));

**if** (newString == **NULL**)

{

**return** **NULL**;

}

**else**

{

strcpy(newString, s->num);

strcat(newString, ".");

strcat(newString, s->frac);

newString[count1+count2+2] = 0;

}

**return** newString;

}

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שאלה 3

**void** real\_add\_as\_String(Real \*numToAdd, Real \*base)

{

**int** numAdd = 0, fracAdd = 0, numBase = 0, fracBase = 0, TempFracAdd, TempfracBase;

**int** countDigits = 0, countDigits1 = 0, countmax = 0, countShalem = 0, countSherit = 0;

**int** tempNumAdd = 0;

**int** j = 0, index = 0, modolo = 0;

**while** (numToAdd->num[j] != 0)

{

countDigits++;

j++;

}

**for** (**int** i = countDigits - 1; i >= 0; i--)

{

numAdd = (numAdd \* 10) + ((**int**)numToAdd->num[index] - 48);

index++;

}

countDigits = 0;

j = 0;

index = 0;

**while** (numToAdd->frac[j] != 0)

{

countDigits++;

j++;

}

**for** (**int** i = countDigits - 1; i >= 0; i--)

{

fracAdd = (fracAdd \* 10) + ((**int**)numToAdd->frac[index] - 48);

index++;

}

countDigits = 0;

j = 0;

index = 0;

**while** (base->num[j] != 0)

{

countDigits++;

j++;

}

**for** (**int** i = countDigits - 1; i >= 0; i--)

{

numBase = (numBase \* 10) + ((**int**)base->num[index] - 48);

index++;

}

countDigits = 0;

j = 0;

index = 0;

**while** (base->frac[j] != 0)

{

countDigits++;

j++;

}

**for** (**int** i = countDigits - 1; i >= 0; i--)

{

fracBase = (fracBase \* 10) + ((**int**)base->frac[index] - 48);

index++;

}

countDigits = 0;

numAdd = numAdd + numBase;

TempFracAdd = fracAdd;

TempfracBase = fracBase;

**while**(fracAdd != 0)

{

fracAdd = fracAdd / 10;

countDigits++;

}

**while**(fracBase != 0)

{

fracBase = fracBase / 10;

countDigits1++;

}

**if** (countDigits > countDigits1)

{

countmax = pow(10, countDigits);

TempfracBase = TempfracBase \* (pow(10,  
 (countDigits - countDigits1)));  
 countSherit = countDigits;

}

if(countDigits1 > countDigits)

{

countmax = pow(10, countDigits1);

TempFracAdd = TempFracAdd \* (pow(10,(countDigits1 - countDigits)));

countSherit = countDigits1;

}

if(countDigits == countDigits1)

{

countmax = pow(10, countDigits1);

countSherit = countDigits;

}

fracAdd = TempFracAdd + TempfracBase;

**if** (fracAdd > countmax)

{

modolo = fracAdd % countmax;

numAdd++;

fracAdd = modolo;

}

tempNumAdd = numAdd;

**while** (numAdd != 0)

{

numAdd = numAdd / 10;

countShalem++;

}

base->num = (**char**\*)malloc((countShalem + 1) \* **sizeof**(**char**));

base->frac = (**char**\*)malloc((countSherit + 1) \* **sizeof**(**char**));

**if** (base->num == **NULL**)

{

**return**;

}

**for** (**char**\* p = base->num + countShalem - 1; p >= base->num; p--)

{

\*p = (**char**)(tempNumAdd % 10+48);

tempNumAdd = tempNumAdd / 10;

}

base->num[countShalem] = 0;

**for** (**char**\* p = base->frac + countSherit - 1; p >= base->frac; p--)

{

\*p = (**char**)(fracAdd % 10+48);

fracAdd = fracAdd / 10;

}

base->frac[countSherit] = 0;

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי}