עבודת הגשה -01

שמעון דסטה 203670286
טל סער 209151380

<u>הפתרון לתרגילים:</u>

שאלה 1:

```
void printArray(char* arr, int size)
{
	for (int i = 0; i < size; i++)
	{
		printf("%c", *(arr + i));
	}
	printf("\n");
}

void merge_lists(char list1[], char list2[], char list3[], int m, int n)
{
	if (m != 1)
	{
		*list3 = *list1;
	}
	else
	{
		for (int i = 0; i < n - 1; i++)
		{
			*(list3 + i) = (*list2 + i);
		}
		return;
	}
	if (n != 1)
	{
		*(list3 + 1) = *list2;
	}
	else
	{
		for (int i = 0; i < m - 1; i++)
		{
			*(list3 + i) = *(list1 + i);
		}
		return;
	}
	merge_lists(list1 + 1, list2 + 1, list3 + 2, m - 1, n - 1);
}
```

שאלה 2:

סעיף א. - 1
סעיף ב. - 2
סעיף ג. - 3
סעיף ד: - 4

שאלה 3:

```
1. n1 == n2 ? return 1 : return 0
2. what(n1, n2 / 10)
3 what (n1 / 10, n2)
4. what(n1 / 10, n2 / 10)
```

שאלה 4:

```
int decimal_base(int num)
{
   if (num == 1)
   {
      return 1;
   }
   return decimal_base(num / 2) * 10 + num % 2;
}
```

שאלה 5:

```
1. dig2 = (char)((dig1) + 48);
2. dig2 = (char)((dig1)+55);
3. makenumintohexadecimal(num/16);
4. int len = strlen(s)
5. *(s + len) = dig2;
6. *(s + len + 1) = '\0';
```

שאלה 6:

```
#include <stdio.h>

int ternarySearch(int arr[], int low, int hi, int num)
{
        if (hi >= low)
        {
                int mid1 = low + (hi - low) / 3;
                int mid2 = hi - (hi - low) / 3;

                if (arr[mid1] == num)
                        return mid1;
                if (arr[mid2] == num)
                        return mid2;
                if (num < arr[mid1])
                        return ternarySearch(arr, low, mid1 - 1, num);
                else if (num > arr[mid2])
                        return ternarySearch(arr, mid2 + 1, hi, num);
                else
                        return ternarySearch(arr, mid1 + 1, mid2 - 1, num);
        }
        else
                return -1;
}

int main(void)
{

        int arr[20] ;
        int result, key, size;
        int count = 0;

        printf("Enter the size of the list: ");
        scanf("%d", &size);
        printf("Enter a %d numbers to list:\n",size);
        printf("The random list:\n");
        for (int i = 0; i < size; i++) {
                arr[i] = rand() % size;
                printf("%d   ", arr[i]);
        }
        printf("\nEnter the key to search: ");
        scanf("%d", &key);
        result = ternarySearch(arr, 0, size, key);
        if(result == -1){
                printf("Key not found.\n");
                }
        else
                printf("Result is %d\n", result);

        return 0;
}
```

שאלה 7:

f(m, n) = 0.5(f(m + 1, n - 1) + f(m - 1, n + 1)), stop conditions: f(0,n)=1 , f(m,0)=0

שאלה 8:

1. n == index
2. *(word + index) = 0;
3. bin(word, n, index + 1);
4. *(word + index) = 1;

שאלה 9:

```
#include <stdio.h>

void TR8(int* word, int size, int b, int index);
int main() {
        int n, b, index = 0;

        printf("Please enter the length: ");
        scanf("%d", &n);
        printf("Please enter the size: ");
        scanf("%d", &b);

        int* word = (int*)malloc(n * sizeof(int));
        TR8(word, n, b, index);
        free(word);
        return 0;
}




void TR8(int* word,  int n, int b, int index)
```

```
{
	if (n == index)
	{
		for (int i = 0; i < n; i++)
		{
			printf("%d ", *(word + i));
		}
		printf("\n");
		return;
	}
	for (int i = 0; i < b; i++)
	{
		*(word + index) = i;
		TR8(word, n, b, index + 1);
	}
}
```

שאלה 10:

```
#include <stdio.h>


void printArray(int *arr, int m);
void printC(int *arr, int n, int i);
int main(void)
{
	int n ;
	printf("Enter a number:\n");
	scanf("%d", &n);
	int* arr = (int*)malloc(n * sizeof(int));
	printC(arr, n, 0);

	return 0;
}
void printArray(int *arr, int m)
{
	for (int i = 0; i < m; i++) {
		printf("%d ", arr[i]);
	}
	printf("\n");
}

void printC(int *arr, int n, int i){

	if (n == 0) {
		printArray(arr, i);
	}
	else if (n > 0) {
		for (int k = 1; k <= n; k++) {
			arr[i] = k;
			printC(arr, n - k, i + 1);
		}

	}
}
```

שאלה 11:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>


#define N  8
#define DIAG  (N*2 - 1)
#define BOARD_SIDE 8

void TR_Solutionsr(int board[N], int column, int used[3][DIAG]);
void SolutionPrint(int board[N], int size);
void UsedAdd(int used[3][DIAG], int row, int column);
void UsedRemove(int used[3][DIAG], int row, int column);
bool Legal(int used[3][DIAG], int row, int column);

int main()
{
	int board[N];
	int used[3][DIAG]; // row, up-diag, down-diag

	//Initializing used arrays. Note that there are
	//only N rows so the remaining elements
	//in used[0] will be discarded.

	for (int i = 0; i < DIAG; i++)
		used[0][i] = used[1][i] = used[2][i] = false;

	printf("Solving the %d Queens Problem\n\n", N);
	TR_Solutionsr(board, 0, used);
	printf("\nDone!\n");

	return 0;
}








void TR_Solutionsr(int board[N], int column, int used[3][DIAG])
```

```
{
	if (column >= N) {
		SolutionPrint(board, N);
		return;
	}

	for (int row = 0; row < N; row++) {
		if (!Legal(used, row, column))
			continue;
		board[column] = row;
		UsedAdd(used, row, column);
		TR_Solutionsr(board, column + 1, used);
		UsedRemove(used, row, column);
	}

	return;
}

bool Legal(int used[3][DIAG], int r, int c)
{
	return  !used[0][r]
		&& !used[1][r + c]
		&& !used[2][N - 1 + r - c];
}

void UsedMark(int used[3][DIAG], int mark, int r, int c)
{
	used[0][r] = mark;
	used[1][r + c] = mark;
	used[2][BOARD_SIDE - 1 + r - c] = mark;
}

void UsedAdd(int used[3][DIAG], int r, int c)
{
	UsedMark(used, true, r, c);
}

void UsedRemove(int used[3][DIAG], int r, int c)
{
	UsedMark(used, false, r, c);
}

void SolutionPrint(int board[N], int size)
{
	static int solution_number;
	printf("Solution number %d:\n\t", ++solution_number);
	for (int i = 0; i < size; i++)
		printf("%d ", board[i]);
	putchar('\n');
}
```