ASSIGNMENT 3

1)

CODE:

[*] ASSIGNMENT 3.c

```c
#include <stdio.h>
int reverse(int num) {
    static int revNum = 0;
    int rem;
    if (num != 0) {
        rem = num % 10;
        revNum = revNum * 10 + rem;
        reverse(num / 10);
    } else {
        return revNum;
    }
}
int main() {
    int num, revNum;
    printf("Enter a number: ");
    scanf("%d", &num);
    revNum = reverse(num);
    printf("The reverse of %d is: %d\n", num, revNum);
    return 0;
}
```

OUTPUT:

C:\Users\selco\OneDrive\Doc

```
Enter a number: 3467
The reverse of 3467 is: 7643

--------------------------------
Process exited after 10.71 seconds with return value 0
Press any key to continue . . .
```

2)

CODE:

[*] ASSIGNMENT 3.c

```c
#include <stdio.h>
#include <string.h>
void leftShift(char* s, int amount) {
    int len = strlen(s);
    amount = amount % len;
    char temp[amount + 1];
    strncpy(temp, s, amount);
    temp[amount] = '\0';
    strcpy(s, s + amount);
    strcat(s, temp);
}
void rightShift(char* s, int amount) {
    int len = strlen(s);
    amount = amount % len;
    char temp[len - amount + 1];
    strncpy(temp, s + len - amount, amount);
    temp[amount] = '\0';
    strcpy(s + len - amount, s);
    strncpy(s, temp, amount);
    s[amount] = '\0';
    strcat(s, s + amount);
}
```

[*] ASSIGNMENT 3.c

```c
char* stringShift(char* s, int** shift, int shiftSize, int* shiftColSize) {
    for (int i = 0; i < shiftSize; i++) {
        if (shift[i][0] == 0) {
            leftShift(s, shift[i][1]);
        } else {
            rightShift(s, shift[i][1]);
        }
    }
    return s;
}
int main() {
    char s[] = "abc";
    int shift[][2] = {{0, 1}, {1, 2}};
    int shiftSize = sizeof(shift) / sizeof(shift[0]);
    int shiftColSize = sizeof(shift[0]) / sizeof(shift[0][0]);
    printf("Output: %s\n", stringShift(s, (int**)shift, shiftSize, &shiftColSize));
    char s2[] = "abcdefg";
    int shift2[][2] = {{1, 1}, {1, 1}, {0, 2}, {1, 3}};
    int shiftSize2 = sizeof(shift2) / sizeof(shift2[0]);
    int shiftColSize2 = sizeof(shift2[0]) / sizeof(shift2[0][0]);
    printf("Output: %s\n", stringShift(s2, (int**)shift2, shiftSize2, &shiftColSize2));
    return 0;
}
```

OUTPUT:

C:\Users\selco\OneDrive\Doc

```
--------------------------------
Process exited after 12.58 seconds with return value 3221225477
Press any key to continue . . .
```

3)

CODE:

[*] ASSIGNMENT 3.c

```c
#include <stdio.h>
typedef struct {
    int** matrix;
    int rows;
    int cols;
} BinaryMatrix;
int get(BinaryMatrix* binaryMatrix, int row, int col) {
    return binaryMatrix->matrix[row][col];
}
int* dimensions(BinaryMatrix* binaryMatrix) {
    static int dims[2];
    dims[0] = binaryMatrix->rows;
    dims[1] = binaryMatrix->cols;
    return dims;
}
int leftMostColumnWithOne(BinaryMatrix* binaryMatrix) {
    int rows = binaryMatrix->rows;
    int cols = binaryMatrix->cols;
    int low = 0;
    int high = cols - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;
```

Compile Log | Debug | Find Results | Close

```c
        int i;
        for (i = 0; i < rows; i++) {
            if (get(binaryMatrix, i, mid) == 1) {
                hasOne = 1;
                break;
            }
        }

        if (hasOne) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    int i;
    for ( i = 0; i < rows; i++) {
        if (get(binaryMatrix, i, low) == 1) {
            return low;
        }
    }
    return -1;
}
int main() {
```


```c
    for ( i = 0; i < rows; i++) {
        if (get(binaryMatrix, i, low) == 1) {
            return low;
        }
    }
    return -1;
}
```


```c
int main() {
    int matrix[][5] = {{0, 0, 0, 1, 1}, {0, 0, 1, 1, 1}, {0, 0, 0, 0, 0}, {0, 1, 1, 1, 1}, {0, 0, 0, 1, 1}};
    BinaryMatrix binaryMatrix;
    binaryMatrix.matrix = (int**)matrix;
    binaryMatrix.rows = sizeof(matrix) / sizeof(matrix[0]);
    binaryMatrix.cols = sizeof(matrix[0]) / sizeof(matrix[0][0]);
    printf("Output: %d\n", leftMostColumnWithOne(&binaryMatrix));
    return 0;
}
```

OUTPUT:

```
C:\Users\selco\OneDrive\Doc

--------------------------------
Process exited after 10.73 seconds with return value 3221225477
Press any key to continue . . .
```

4)

CODE:

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int val;
    struct Node* next;
    struct Node* prev;
} Node;

typedef struct {
    Node* head;
    Node* tail;
    int count[1001]; // assuming the maximum value is 1000
} FirstUnique;

FirstUnique* firstUniqueCreate(int* nums, int numsSize) {
    FirstUnique* fu = (FirstUnique*)malloc(sizeof(FirstUnique));
    fu->head = NULL;
    fu->tail = NULL;
    int i;
    for (i = 0; i < numsSize; i++) {
        fu->count[nums[i]]++;
        Node* node = (Node*)malloc(sizeof(Node));
```

Compile Log Debug Find Results Close

ASSIGNMENT 3.c

```c
        fu->count[nums[i]]++;
        Node* node = (Node*)malloc(sizeof(Node));
        node->val = nums[i];
        node->next = NULL;
        node->prev = fu->tail;

        if (fu->head == NULL) {
            fu->head = node;
        } else {
            fu->tail->next = node;
        }
        fu->tail = node;
    }

    return fu;
}

int firstUniqueShowFirstUnique(FirstUnique* fu) {
    Node* node = fu->head;
    while (node!= NULL) {
        if (fu->count[node->val] == 1) {
            return node->val;
        }
```

Compile Log Debug Find Results Close

- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe

ASSIGNMENT 3.c

```c
        if (fu->count[node->val] == 1) {
            return node->val;
        }
        node = node->next;
    }
    return -1;
}

void firstUniqueAdd(FirstUnique* fu, int value) {
    fu->count[value]++;
    Node* node = (Node*)malloc(sizeof(Node));
    node->val = value;
    node->next = NULL;
    node->prev = fu->tail;

    if (fu->head == NULL) {
        fu->head = node;
    } else {
        fu->tail->next = node;
    }
    fu->tail = node;
}
```

OUTPUT:

```
C:\Users\selco\OneDrive\Doc

Output: 2
Output: 2
Output: 3
Output: -1

--------------------------------
Process exited after 4.066 seconds with return value 0
Press any key to continue . . .
```

5)

CODE:

ASSIGNMENT 3.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct TreeNode {
    int val;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;

int isValidSequence(TreeNode* root, int* arr, int arrSize, char* path, int pathLen) {
    if (root == NULL) {
        return 0;
    }

    char str[100];
    sprintf(str, "%d", root->val);
    strcat(path, str);

    if (root->left == NULL && root->right == NULL) {
        int len = strlen(path);
        int i;
        for (i = 0; i < arrSize; i++) {
```

```c
        int len = strlen(path);
        int i;
        for (i = 0; i < arrSize; i++) {
            char str[100];
            sprintf(str, "%d", arr[i]);
            if (strncmp(path, str, strlen(str)) != 0) {
                return 0;
            }
            path += strlen(str);
        }
        return 1;
    }

    int res = 0;
    if (root->left != NULL) {
        res |= isValidSequence(root->left, arr, arrSize, path, pathLen);
    }
    if (root->right != NULL) {
        res |= isValidSequence(root->right, arr, arrSize, path, pathLen);
    }

    path[pathLen - strlen(str)] = '\0';
    return res;
```

Compile Log Debug Find Results Close

ASSIGNMENT 3.c

```c

    path[pathLen - strlen(str)] = '\0';
    return res;
}

int isValidSequenceFromRootToLeaves(TreeNode* root, int* arr, int arrSize) {
    char path[100] = "";
    return isValidSequence(root, arr, arrSize, path, 0);
}

int main() {
    TreeNode* root = (TreeNode*)malloc(sizeof(TreeNode));
    root->val = 0;
    root->left = (TreeNode*)malloc(sizeof(TreeNode));
    root->left->val = 1;
    root->right = (TreeNode*)malloc(sizeof(TreeNode));
    root->right->val = 0;
    root->left->left = (TreeNode*)malloc(sizeof(TreeNode));
    root->left->left->val = 0;
    root->left->right = (TreeNode*)malloc(sizeof(TreeNode));
    root->left->right->val = 1;
    root->right->left = (TreeNode*)malloc(sizeof(TreeNode));
    root->right->left->val = 0;
```

```c
    root->left->right = (TreeNode*)malloc(sizeof(TreeNode));
    root->left->right->val = 1;
    root->right->left = (TreeNode*)malloc(sizeof(TreeNode));
    root->right->left->val = 0;
    root->right->right = (TreeNode*)malloc(sizeof(TreeNode));
    root->right->right->val = 1;
    root->left->left->left = NULL;
    root->left->left->right = NULL;
    root->left->right->left = NULL;
    root->left->right->right = NULL;
    root->right->left->left = NULL;
    root->right->left->right = NULL;
    root->right->right->left = NULL;
    root->right->right->right = NULL;

    int arr[] = {0, 1, 0, 1};
    int arrSize = sizeof(arr) / sizeof(arr[0]);

    printf("Output: %d\n", isValidSequenceFromRootToLeaves(root, arr, arrSize)); // return 1

    return 0;
}
```

Compile Log Debug Find Results Close

OUTPUT:

```
C:\Users\selco\OneDrive\Doc

Output: 1

--------------------------------
Process exited after 2.099 seconds with return value 0
Press any key to continue . . .
```

6)

CODE:

[*] ASSIGNMENT 3.c

```c
#include <stdio.h>
#include <stdlib.h>
int* kidsWithCandies(int* candies, int candiesSize, int extraCandies, int* returnSize) {
    int maxCandies = 0;
    for (int i = 0; i < candiesSize; i++) {
        if (candies[i] > maxCandies) {
            maxCandies = candies[i];
        }
    }
    int* result = (int*)malloc(candiesSize * sizeof(int));
    for (int i = 0; i < candiesSize; i++) {
        result[i] = (candies[i] + extraCandies) >= maxCandies;
    }
    *returnSize = candiesSize;
    return result;
}
int main() {
    int candies[] = {2, 3, 5, 1, 3};
    int candiesSize = sizeof(candies) / sizeof(candies[0]);
    int extraCandies = 3;
    int returnSize;
    int* result = kidsWithCandies(candies, candiesSize, extraCandies, &returnSize);
    for (int i = 0; i < returnSize; i++) {
```

```c
}
int main() {
    int candies[] = {2, 3, 5, 1, 3};
    int candiesSize = sizeof(candies) / sizeof(candies[0]);
    int extraCandies = 3;
    int returnSize;
    int* result = kidsWithCandies(candies, candiesSize, extraCandies, &returnSize);
    for (int i = 0; i < returnSize; i++) {
        printf("%d ", result[i]);
    }
    printf("\n");
    free(result);
    return 0;
}
```

OUTPUT:

C:\Users\selco\OneDrive\Doc

```
1 1 1 0 1

--------------------------------
Process exited after 2.068 seconds with return value 0
Press any key to continue . . .
```

7)

CODE:

ASSIGNMENT 3.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int maximumGap(int num) {
    char str[10];
    sprintf(str, "%d", num);
    int len = strlen(str);
    int maxDiff = 0;
    int i;
    for (i = 0; i < 10; i++) {
        int j;
        for (j = 0; j < 10; j++) {
            char temp[10];
            strcpy(temp, str);
            int k;
            for (k = 0; k < len; k++) {
                if (temp[k] - '0' == i) {
                    temp[k] = j + '0';
                }
            }
            if (temp[0]!= '0' || len == 1) {
                int a = atoi(temp);
                strcpy(temp, str);
```

ASSIGNMENT 3.c

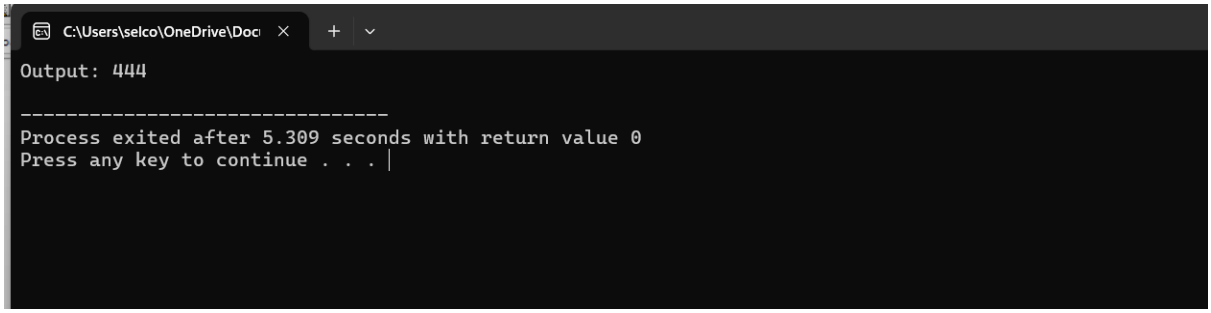
```c
                }
            }
            if (temp[0]!= '0' || len == 1) {
                int a = atoi(temp);
                strcpy(temp, str);
                for (k = 0; k < len; k++) {
                    if (temp[k] - '0' == j) {
                        temp[k] = i + '0';
                    }
                }
                if (temp[0]!= '0' || len == 1) {
                    int b = atoi(temp);
                    maxDiff = maxDiff > abs(a - b)? maxDiff : abs(a - b);
                }
            }
        }
    }
    return maxDiff;
}
int main() {
    int num = 555;
    printf("Output: %d\n", maximumGap(num)); // return 888 - 99 = 789
```

public int __cdecl printf (const char * __restrict__ _Format, ...)

Compile Log | Debug | Find Results | Close

```c
    }
    return maxDiff;
}
int main() {
    int num = 555;
    printf("Output: %d\n", maximumGap(num)); // return 888 - 99 = 789
    return 0;
}
```

OUTPUT:


C:\Users\selco\OneDrive\Doc
Output: 444

Process exited after 5.309 seconds with return value 0
Press any key to continue . . .

8)
CODE:

[*] ASSIGNMENT 3.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int canBreak(char* s1, char* s2, int n) {
    int count[26] = {0};
    for (int i = 0; i < n; i++) {
        count[s1[i] - 'a']++;
        count[s2[i] - 'a']--;
    }
    for (int i = 0; i < 26; i++) {
        if (count[i] > 0) return 0;
    }
    return 1;
}
int main() {
    char s1[] = "abc";
    char s2[] = "xya";
    int n = strlen(s1);
    if (canBreak(s1, s2, n) || canBreak(s2, s1, n)) {
        printf("True\n");
    } else {
        printf("False\n");
    }
    return 0;
}
```

urces Compile Log Debug Find Results Close

OUTPUT:

C:\Users\selco\OneDrive\Doc

```
False

--------------------------------
Process exited after 2.074 seconds with return value 0
Press any key to continue . . .
```

9)

CODE:

[*] ASSIGNMENT 3.c

```c
#include <stdio.h>
#include <stdlib.h>
#define MOD 1000000007
int countWays(int*** dp, int n, int hats[][41], int mask, int idx) {
    if (idx == n) return 1;
    if (dp[idx][mask]!= -1) return dp[idx][mask];
    int ways = 0;
    ways = (ways + countWays(dp, n, hats, mask, idx + 1)) % MOD;
    for (int i = 0; i < 41; i++) {
        if (hats[idx][i] &&!(mask & (1 << i))) {
            ways = (ways + countWays(dp, n, hats, mask | (1 << i), idx + 1)) % MOD;
        }
    }
    dp[idx][mask] = ways;
    return ways;
}
int numberWays(int** hats, int hatsSize, int* hatsColSize, int n) {
    int** dp = (int**)malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++) {
        dp[i] = (int*)calloc(1 << 40, sizeof(int));
        for (int j = 0; j < (1 << 40); j++) {
            dp[i][j] = -1;
        }
```

[*] ASSIGNMENT 3.c

```c
    }
    int hatsArray[n][41];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 41; j++) {
            hatsArray[i][j] = 0;
        }
    }
    for (int i = 0; i < hatsSize; i++) {
        for (int j = 0; j < hatsColSize[i]; j++) {
            hatsArray[i][hats[i][j]] = 1;
        }
    }
    return countWays((int***)dp, n, hatsArray, 0, 0);
}
int main() {
    int hats[][41] = {{3, 4}, {4, 5}, {5}};
    int hatsSize = sizeof(hats) / sizeof(hats[0]);
    int hatsColSize[] = {2, 2, 1};
    int n = sizeof(hats) / sizeof(hats[0]);
    printf("Output: %d\n", numberWays((int**)hats, hatsSize, hatsColSize, n));
    public int __cdecl printf (const char * __restrict__ _Format, ...)
}
```

OUTPUT:

```
C:\Users\selco\OneDrive\Doc

Output: 0

--------------------------------
Process exited after 2.06 seconds with return value 0
Press any key to continue . . .
```

10)

CODE:

```c
#include <stdio.h>
void nextPermutation(int* nums, int numsSize) {
    int i = numsSize - 2;
    while (i >= 0 && nums[i] >= nums[i + 1]) {
        i--;
    }
    if (i >= 0) {
        int j = numsSize - 1;
        while (j > i && nums[j] <= nums[i]) {
            j--;
        }
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }
    int left = i + 1;
    int right = numsSize - 1;
    while (left < right) {
        int temp = nums[left];
        nums[left] = nums[right];
        nums[right] = temp;
        left++;
        right--;
```

[*] ASSIGNMENT 3.c

```c
        nums[i] = nums[j];
        nums[j] = temp;
    }
    int left = i + 1;
    int right = numsSize - 1;
    while (left < right) {
        int temp = nums[left];
        nums[left] = nums[right];
        nums[right] = temp;
        left++;
        right--;
    }
}
int main() {
    int nums[] = {1, 2, 3};
    int numsSize = sizeof(nums) / sizeof(nums[0]);
    nextPermutation(nums, numsSize);
    for (int i = 0; i < numsSize; i++) {
        printf("%d ", nums[i]);
    }
    return 0;
}
```

OUTPUT:

C:\Users\selco\OneDrive\Doc

```
1 3 2
--------------------------------
Process exited after 2.048 seconds with return value 0
Press any key to continue . . .
```