

ASSIGNMENT 3

1)

CODE:

```
1 #include <stdio.h>
2 int reverse(int num) {
3     static int revNum = 0;
4     int rem;
5     if (num != 0) {
6         rem = num % 10;
7         revNum = revNum * 10 + rem;
8         reverse(num / 10);
9     } else {
10        return revNum;
11    }
12 }
13 int main() {
14     int num, revNum;
15     printf("Enter a number: ");
16     scanf("%d", &num);
17     revNum = reverse(num);
18     printf("The reverse of %d is: %d\n", num, revNum);
19     return 0;
20 }
```

OUTPUT:

```
C:\Users\seldo\OneDrive\Doc  ×  +  v
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
1  #include <stdio.h>
2  #include <string.h>
3  void leftShift(char* s, int amount) {
4      int len = strlen(s);
5      amount = amount % len;
6      char temp[amount + 1];
7      strncpy(temp, s, amount);
8      temp[amount] = '\0';
9      strcpy(s, s + amount);
10     strcat(s, temp);
11 }
12 void rightShift(char* s, int amount) {
13     int len = strlen(s);
14     amount = amount % len;
15     char temp[len - amount + 1];
16     strncpy(temp, s + len - amount, amount);
17     temp[amount] = '\0';
18     strcpy(s + len - amount, s);
19     strncpy(s, temp, amount);
20     s[amount] = '\0';
21     strcat(s, s + amount);
22 }
23
24 char* stringShift(char* s, int** shift, int shiftSize, int* shiftColSize) {
25     for (int i = 0; i < shiftSize; i++) {
26         if (shift[i][0] == 0) {
27             leftShift(s, shift[i][1]);
28         } else {
29             rightShift(s, shift[i][1]);
30         }
31     }
32     return s;
33 }
34
35 int main() {
36     char s[] = "abc";
37     int shift[][2] = {{0, 1}, {1, 2}};
38     int shiftSize = sizeof(shift) / sizeof(shift[0]);
39     int shiftColSize = sizeof(shift[0]) / sizeof(shift[0][0]);
40     printf("Output: %s\n", stringShift(s, (int**)shift, shiftSize, &shiftColSize));
41     char s2[] = "abcdefg";
42     int shift2[][2] = {{1, 1}, {1, 1}, {0, 2}, {1, 3}};
43     int shiftSize2 = sizeof(shift2) / sizeof(shift2[0]);
44     int shiftColSize2 = sizeof(shift2[0]) / sizeof(shift2[0][0]);
45     printf("Output: %s\n", stringShift(s2, (int**)shift2, shiftSize2, &shiftColSize2));
46     return 0;
47 }
```

OUTPUT:

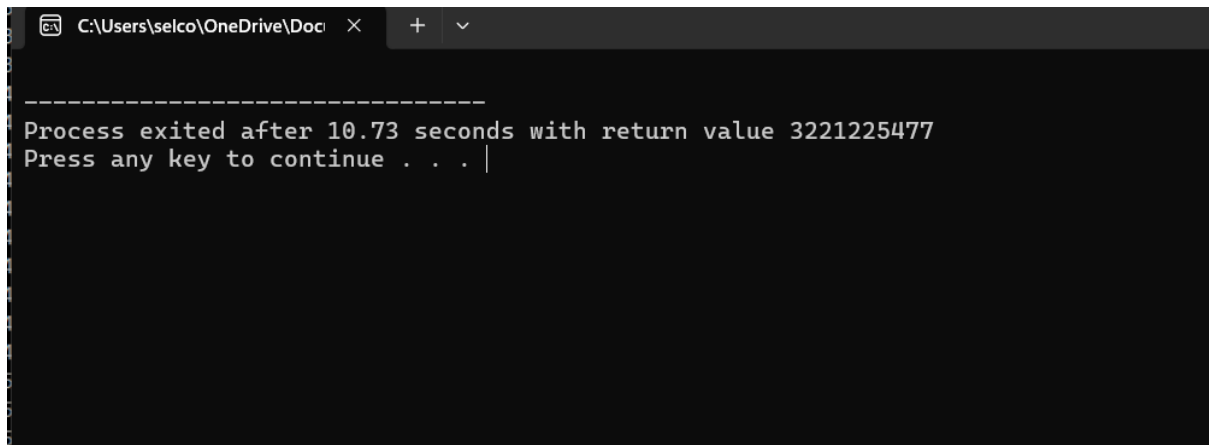
```
C:\Users\seldo\OneDrive\Docu  X  +  v
-----
Process exited after 12.58 seconds with return value 3221225477
Press any key to continue . . . |
```

3)

CODE:

```
1 #include <stdio.h>
2 typedef struct {
3     int** matrix;
4     int rows;
5     int cols;
6 } BinaryMatrix;
7 int get(BinaryMatrix* binaryMatrix, int row, int col) {
8     return binaryMatrix->matrix[row][col];
9 }
10 int* dimensions(BinaryMatrix* binaryMatrix) {
11     static int dims[2];
12     dims[0] = binaryMatrix->rows;
13     dims[1] = binaryMatrix->cols;
14     return dims;
15 }
16 int leftMostColumnWithOne(BinaryMatrix* binaryMatrix) {
17     int rows = binaryMatrix->rows;
18     int cols = binaryMatrix->cols;
19     int low = 0;
20     int high = cols - 1;
21
22     while (low <= high) {
23         int mid = low + (high - low) / 2;
24
25         int i;
26         for (i = 0; i < rows; i++) {
27             if (get(binaryMatrix, i, mid) == 1) {
28                 hasOne = 1;
29                 break;
30             }
31         }
32
33         if (hasOne) {
34             high = mid - 1;
35         } else {
36             low = mid + 1;
37         }
38     }
39     int i;
40     for (i = 0; i < rows; i++) {
41         if (get(binaryMatrix, i, low) == 1) {
42             return low;
43         }
44     }
45     return -1;
46 }
47 int main() {
48     for (i = 0; i < rows; i++) {
49         if (get(binaryMatrix, i, low) == 1) {
50             return low;
51         }
52     }
53     return -1;
54 }
55
56 int main() {
57     int matrix[5][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}};
58     BinaryMatrix binaryMatrix;
59     binaryMatrix.matrix = (int**)matrix;
60     binaryMatrix.rows = sizeof(matrix) / sizeof(matrix[0]);
61     binaryMatrix.cols = sizeof(matrix[0]) / sizeof(matrix[0][0]);
62     printf("Output: %d\n", leftMostColumnWithOne(&binaryMatrix));
63     return 0;
64 }
```

OUTPUT:



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

The image shows a Windows command prompt window. The title bar at the top indicates the current directory is C:\Users\sclco\OneDrive\Doc. The command prompt itself displays a message indicating that a process has exited after 10.73 seconds with a return value of 3221225477. It then prompts the user to press any key to continue, with a cursor visible after the text.

4)

CODE:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct Node {
5      int val;
6      struct Node* next;
7      struct Node* prev;
8  } Node;
9
10 typedef struct {
11     Node* head;
12     Node* tail;
13     int count[1001]; // assuming the maximum value is 1000
14 } FirstUnique;
15
16 FirstUnique* firstUniqueCreate(int* nums, int numsSize) {
17     FirstUnique* fu = (FirstUnique*)malloc(sizeof(FirstUnique));
18     fu->head = NULL;
19     fu->tail = NULL;
20     int i;
21     for (i = 0; i < numsSize; i++) {
22         fu->count[nums[i]]++;
23         Node* node = (Node*)malloc(sizeof(Node));
24
25         fu->count[nums[i]]++;
26         Node* node = (Node*)malloc(sizeof(Node));
27         node->val = nums[i];
28         node->next = NULL;
29         node->prev = fu->tail;
30
31         if (fu->head == NULL) {
32             fu->head = node;
33         } else {
34             fu->tail->next = node;
35         }
36         fu->tail = node;
37     }
38     return fu;
39
40 int firstUniqueShowFirstUnique(FirstUnique* fu) {
41     Node* node = fu->head;
42     while (node != NULL) {
43         if (fu->count[node->val] == 1) {
44             return node->val;
45         }
46         node = node->next;
47     }
48     return -1;
49
50 void firstUniqueAdd(FirstUnique* fu, int value) {
51     fu->count[value]++;
52     Node* node = (Node*)malloc(sizeof(Node));
53     node->val = value;
54     node->next = NULL;
55     node->prev = fu->tail;
56
57     if (fu->head == NULL) {
58         fu->head = node;
59     } else {
60         fu->tail->next = node;
61     }
62     fu->tail = node;
63 }
```

OUTPUT:

```
C:\Users\selco\OneDrive\Doc... X + v
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
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  typedef struct TreeNode {
6      int val;
7      struct TreeNode* left;
8      struct TreeNode* right;
9  } TreeNode;
10
11 int isValidSequence(TreeNode* root, int* arr, int arrSize, char* path, int pathLen) {
12     if (root == NULL) {
13         return 0;
14     }
15
16     char str[100];
17     sprintf(str, "%d", root->val);
18     strcat(path, str);
19
20     if (root->left == NULL && root->right == NULL) {
21         int len = strlen(path);
22         int i;
23         for (i = 0; i < arrSize; i++) {
24             int len = strlen(path);
25             int i;
26             for (i = 0; i < arrSize; i++) {
27                 char str[100];
28                 sprintf(str, "%d", arr[i]);
29                 if (strncmp(path, str, strlen(str)) != 0) {
30                     return 0;
31                 }
32                 path += strlen(str);
33             }
34             return 1;
35         }
36     }
37
38     int res = 0;
39     if (root->left != NULL) {
40         res |= isValidSequence(root->left, arr, arrSize, path, pathLen);
41     }
42     if (root->right != NULL) {
43         res |= isValidSequence(root->right, arr, arrSize, path, pathLen);
44     }
45
46     path[pathLen - strlen(str)] = '\0';
47     return res;
48 }
49
50 int isValidSequenceFromRootToLeaves(TreeNode* root, int* arr, int arrSize) {
51     char path[100] = "";
52     return isValidSequence(root, arr, arrSize, path, 0);
53 }
54
55 int main() {
56     TreeNode* root = (TreeNode*)malloc(sizeof(TreeNode));
57     root->val = 0;
58     root->left = (TreeNode*)malloc(sizeof(TreeNode));
59     root->left->val = 1;
60     root->right = (TreeNode*)malloc(sizeof(TreeNode));
61     root->right->val = 0;
62     root->left->left = (TreeNode*)malloc(sizeof(TreeNode));
63     root->left->left->val = 0;
64     root->left->right = (TreeNode*)malloc(sizeof(TreeNode));
65     root->left->right->val = 1;
66     root->right->left = (TreeNode*)malloc(sizeof(TreeNode));
67     root->right->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;
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\seldo\OneDrive\Doc x + v

Output: 1

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


6)

CODE:

[*] ASSIGNMENT 3.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int* kidsWithCandies(int* candies, int candiesSize, int extraCandies, int* returnSize) {
4      int maxCandies = 0;
5      for (int i = 0; i < candiesSize; i++) {
6          if (candies[i] > maxCandies) {
7              maxCandies = candies[i];
8          }
9      }
10     int* result = (int*)malloc(candiesSize * sizeof(int));
11     for (int i = 0; i < candiesSize; i++) {
12         result[i] = (candies[i] + extraCandies) >= maxCandies;
13     }
14     *returnSize = candiesSize;
15     return result;
16 }
17 int main() {
18     int candies[] = {2, 3, 5, 1, 3};
19     int candiesSize = sizeof(candies) / sizeof(candies[0]);
20     int extraCandies = 3;
21     int returnSize;
22     int* result = kidsWithCandies(candies, candiesSize, extraCandies, &returnSize);
23     for (int i = 0; i < returnSize; i++) {
24         printf("%d ", result[i]);
25     }
26     printf("\n");
27     free(result);
28     return 0;
29 }
```

OUTPUT:

```
C:\Users\seldo\OneDrive\Docu X + v
1 1 1 0 1

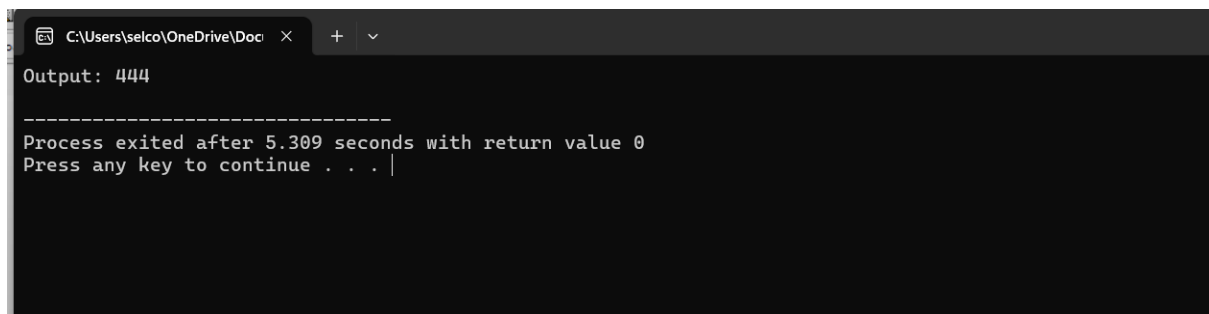
-----
Process exited after 2.068 seconds with return value 0
Press any key to continue . . .
```

7)

CODE:

```
ASSIGNMENT 3.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  int maximumGap(int num) {
5      char str[10];
6      sprintf(str, "%d", num);
7      int len = strlen(str);
8      int maxDiff = 0;
9      int i;
10     for (i = 0; i < 10; i++) {
11         int j;
12         for (j = 0; j < 10; j++) {
13             char temp[10];
14             strcpy(temp, str);
15             int k;
16             for (k = 0; k < len; k++) {
17                 if (temp[k] - '0' == i) {
18                     temp[k] = j + '0';
19                 }
20             }
21             if (temp[0] != '0' || len == 1) {
22                 int a = atoi(temp);
23                 strcpy(temp, str);
24             }
25             if (temp[0] != '0' || len == 1) {
26                 int a = atoi(temp);
27                 strcpy(temp, str);
28                 for (k = 0; k < len; k++) {
29                     if (temp[k] - '0' == j) {
30                         temp[k] = i + '0';
31                     }
32                 }
33                 if (temp[0] != '0' || len == 1) {
34                     int b = atoi(temp);
35                     maxDiff = maxDiff > abs(a - b) ? maxDiff : abs(a - b);
36                 }
37             }
38         }
39     }
40     return maxDiff;
41 }
42
43 int main() {
44     int num = 555;
45     printf("Output: %d\n", maximumGap(num)); // return 888 - 99 = 789
46     public int __cdecl printf (const char * __restrict __Format, ...)
47 }
48
49 }
50 return maxDiff;
51 }
52 int main() {
53     int num = 555;
54     printf("Output: %d\n", maximumGap(num)); // return 888 - 99 = 789
55     return 0;
56 }
```

OUTPUT:



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\sclco\OneDrive\Doc" and standard window controls. The command prompt displays the output "Output: 444". Below this, a separator line of dashes is shown. The final output lines are "Process exited after 5.309 seconds with return value 0" and "Press any key to continue . . . |", where the vertical bar indicates the cursor position.

```
Output: 444

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

8)
CODE:

```
[*] ASSIGNMENT 3.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  int canBreak(char* s1, char* s2, int n) {
5      int count[26] = {0};
6      for (int i = 0; i < n; i++) {
7          count[s1[i] - 'a']++;
8          count[s2[i] - 'a']--;
9      }
10     for (int i = 0; i < 26; i++) {
11         if (count[i] > 0) return 0;
12     }
13     return 1;
14 }
15 int main() {
16     char s1[] = "abc";
17     char s2[] = "xya";
18     int n = strlen(s1);
19     if (canBreak(s1, s2, n) || canBreak(s2, s1, n)) {
20         printf("True\n");
21     } else {
22         printf("False\n");
23     }
24 }
25
```

urces Compile Log Debug Find Results Close

OUTPUT:

```
C:\Users\selsco\OneDrive\Doc x + v
False

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

9)
CODE:

```

[*] ASSIGNMENT 3.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define MOD 1000000007
4  int countWays(int*** dp, int n, int hats[][41], int mask, int idx) {
5      if (idx == n) return 1;
6      if (dp[idx][mask] != -1) return dp[idx][mask];
7      int ways = 0;
8      ways = (ways + countWays(dp, n, hats, mask, idx + 1)) % MOD;
9      for (int i = 0; i < 41; i++) {
10         if (hats[idx][i] &&!(mask & (1 << i))) {
11             ways = (ways + countWays(dp, n, hats, mask | (1 << i), idx + 1)) % MOD;
12         }
13     }
14     dp[idx][mask] = ways;
15     return ways;
16 }
17 int numberWays(int** hats, int hatsSize, int* hatsColSize, int n) {
18     int** dp = (int**)malloc(n * sizeof(int*));
19     for (int i = 0; i < n; i++) {
20         dp[i] = (int*)calloc(1 << 40, sizeof(int));
21         for (int j = 0; j < (1 << 40); j++) {
22             dp[i][j] = -1;
23         }
24     }
25     int hatsArray[n][41];
26     for (int i = 0; i < n; i++) {
27         for (int j = 0; j < 41; j++) {
28             hatsArray[i][j] = 0;
29         }
30     }
31     for (int i = 0; i < hatsSize; i++) {
32         for (int j = 0; j < hatsColSize[i]; j++) {
33             hatsArray[i][hats[i][j]] = 1;
34         }
35     }
36     return countWays((int***)dp, n, hatsArray, 0, 0);
37 }
38 int main() {
39     int hats[][41] = {{3, 4}, {4, 5}, {5}};
40     int hatsSize = sizeof(hats) / sizeof(hats[0]);
41     int hatsColSize[] = {2, 2, 1};
42     int n = sizeof(hats) / sizeof(hats[0]);
43     printf("Output: %d\n", numberWays((int**)hats, hatsSize, hatsColSize, n));
44     public int __cdecl printf (const char * __restrict __Format, ...)
45 }

```

OUTPUT:

```

C:\Users\selsco\OneDrive\Docu X + v
Output: 0
-----
Process exited after 2.06 seconds with return value 0
Press any key to continue . . .

```

10)
CODE:

```

1 #include <stdio.h>
2 void nextPermutation(int* nums, int numsSize) {
3     int i = numsSize - 2;
4     while (i >= 0 && nums[i] >= nums[i + 1]) {
5         i--;
6     }
7     if (i >= 0) {
8         int j = numsSize - 1;
9         while (j > i && nums[j] <= nums[i]) {
10             j--;
11         }
12         int temp = nums[i];
13         nums[i] = nums[j];
14         nums[j] = temp;
15     }
16     int left = i + 1;
17     int right = numsSize - 1;
18     while (left < right) {
19         int temp = nums[left];
20         nums[left] = nums[right];
21         nums[right] = temp;
22         left++;
23         right--;

```

```

13         nums[i] = nums[j];
14         nums[j] = temp;
15     }
16     int left = i + 1;
17     int right = numsSize - 1;
18     while (left < right) {
19         int temp = nums[left];
20         nums[left] = nums[right];
21         nums[right] = temp;
22         left++;
23         right--;
24     }
25 }
26 int main() {
27     int nums[] = {1, 2, 3};
28     int numsSize = sizeof(nums) / sizeof(nums[0]);
29     nextPermutation(nums, numsSize);
30     for (int i = 0; i < numsSize; i++) {
31         printf("%d ", nums[i]);
32     }
33     return 0;
34 }

```

OUTPUT:

```

C:\Users\seldo\OneDrive\Doc  x  +  v
1 3 2
-----
Process exited after 2.048 seconds with return value 0
Press any key to continue . . . |

```