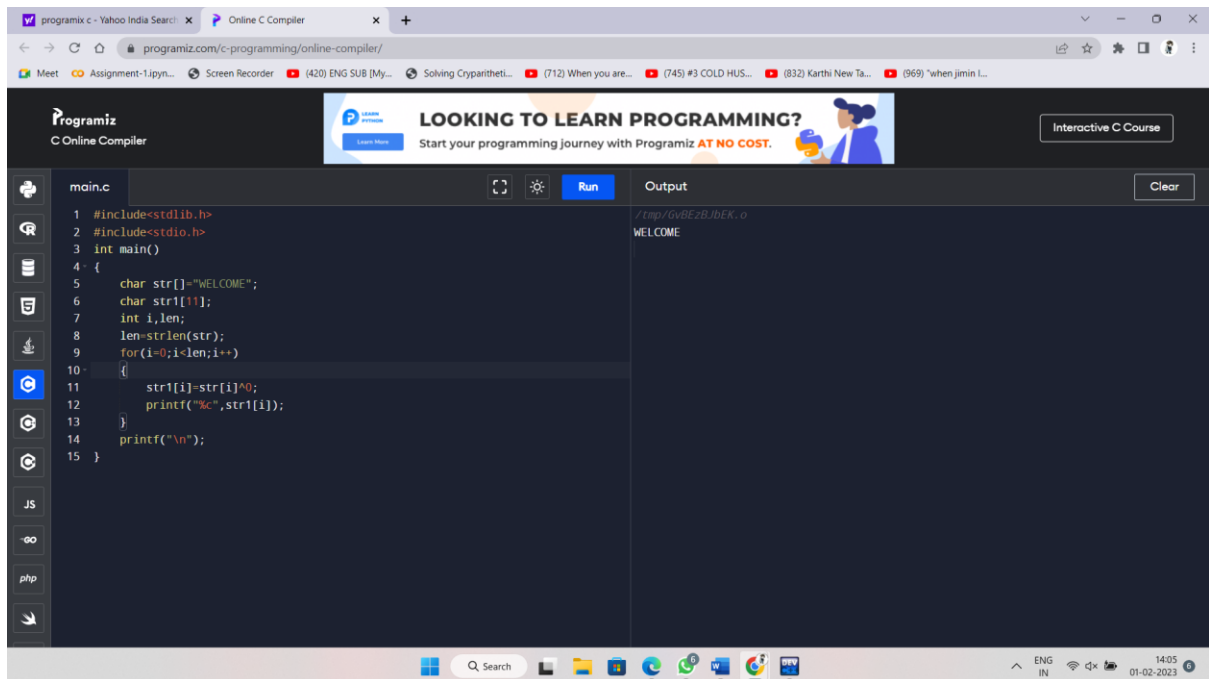


CNS-Programs

R. Aishwarya

192125001

1. Print the elements using the XOR.

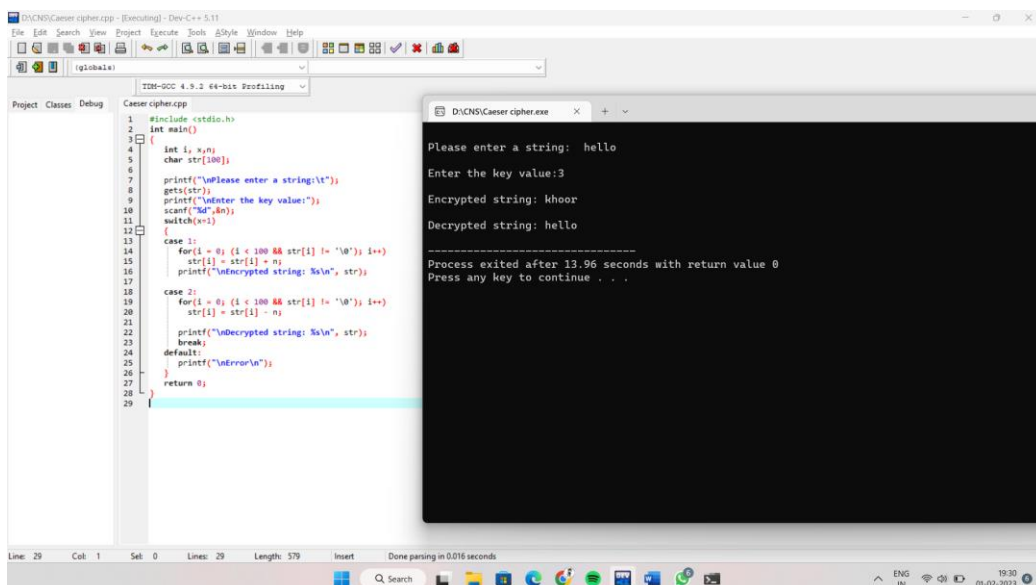


The screenshot shows a web browser with the URL `programiz.com/c-programming/online-compiler/`. The page has a dark theme. At the top, there's a banner for "Programiz C Online Compiler" with a "LOOKING TO LEARN PROGRAMMING?" message and a "Start your programming journey with Programiz AT NO COST." button. Below the banner, there's a sidebar with icons for various programming languages (C, C++, Java, JavaScript, PHP, Python, etc.). The main area is divided into two panels: "main.c" on the left and "Output" on the right. The "main.c" panel contains the following code:

```
1 #include<stdlib.h>
2 #include<stdio.h>
3 int main()
4 {
5     char str[]="WELCOME";
6     char str1[11];
7     int i,len;
8     len=strlen(str);
9     for(i=0;i<len;i++)
10     {
11         str1[i]=str[i]^0;
12         printf("%c",str1[i]);
13     }
14     printf("\n");
15 }
```

The "Output" panel shows the result of the program execution: `/tmp/6vBEzBjBEK.o` and `WELCOME`. The bottom of the browser shows a Windows taskbar with the date and time as 14:05 on 01-02-2023.

2. Write a C program for Caesar cipher involves replacing each letter of the alphabet with the letter stan places further down the alphabet, for k in the range 1 through 25.



The screenshot shows a Windows desktop with a Dev-C++ IDE. The IDE has a menu bar (File, Edit, Search, View, Project, Execute, Tools, Style, Window, Help) and a toolbar. The main window displays a C program for Caesar cipher. The code is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, n;
6     char str[100];
7
8     printf("\nPlease enter a string:\n");
9     gets(str);
10    printf("\nEnter the key value:");
11    scanf("%d",&n);
12    switch(n%26)
13    {
14        case 1:
15            for(i = 0; i < 100 && str[i] != '\0'; i++)
16                str[i] = str[i] + n;
17            printf("\nEncrypted string: %s\n", str);
18        case 2:
19            for(i = 0; i < 100 && str[i] != '\0'; i++)
20                str[i] = str[i] - n;
21            printf("\nDecrypted string: %s\n", str);
22            break;
23        default:
24            printf("\nError\n");
25    }
26    return 0;
27 }
```

The output window shows the execution of the program. It prompts the user to enter a string and a key value. The user enters "hello" and "3". The program outputs the encrypted string "khoor" and the decrypted string "hello". The output window also shows the process exit message: "Process exited after 13.96 seconds with return value 0. Press any key to continue . . .". The bottom of the IDE shows a Windows taskbar with the date and time as 19:30 on 01-02-2023.

3. . Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.

The screenshot shows a C++ IDE with a file named `playfair.cpp`. The code implements the Playfair cipher algorithm. It includes headers for `stdio.h`, `stdlib.h`, and `string.h`. It defines a constant `SIZE` as 30. The `void toLowerCase(char plain[], int ps)` function converts the plaintext to lowercase. The `int removeSpaces(char* plain, int ps)` function removes spaces from the plaintext. The `void generateKeyTable(char key[], int ks, char keyT[5][5])` function generates a 5x5 key table from the keyword. The main function prompts the user for a key and a plaintext, then calls the `generateKeyTable` and `toLowerCase` functions, and finally prints the ciphertext.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #define SIZE 30
5 void toLowerCase(char plain[], int ps)
6 {
7     int i;
8     for (i = 0; i < ps; i++) {
9         if (plain[i] > 64 && plain[i] < 91)
10             plain[i] += 32;
11     }
12 }
13 int removeSpaces(char* plain, int ps)
14 {
15     int i, count = 0;
16     for (i = 0; i < ps; i++)
17         if (plain[i] != ' ')
18             plain[count++] = plain[i];
19     plain[count] = '\0';
20     return count;
21 }
22 void generateKeyTable(char key[], int ks, char keyT[5][5])
23 {
24     int i, j, k, flag = 0, *dicty;
25     dicty = (int*)calloc(26, sizeof(int));
26     for (i = 0; i < ks; i++)
27         if (key[i] != 'j')
28             dicty[key[i] - 97] = 2;
29     dicty['j' - 97] = 1;
30     i = 0;
31     for (k = 0; k < ks; k++) {
32         if (dicty[key[k] - 97] == 2) {
33             dicty[key[k] - 97] -= 1;
34             keyT[i][j] = key[k];
35             i++;
36             if (j == 5) {
37                 j = 0;
38             }
39         }
40     }
41 }
42
43 int main()
44 {
45     char key[50], plain[50], cipher[50];
46     printf("Enter the key: ");
47     scanf("%s", key);
48     printf("Enter the plain text: ");
49     scanf("%s", plain);
50     toLowerCase(plain);
51     removeSpaces(plain);
52     generateKeyTable(key, strlen(key), keyT);
53     printf("Cipher text: ");
54     encrypt(plain, keyT, cipher);
55     printf("\n");
56     return 0;
57 }

```

The execution output shows the key text as "Monarchy", the plain text as "instruments", and the cipher text as "gatlmzclqrxt". The process exited after 10.66 seconds with return value 0.

4. Write a C program for polyalphabetic substitution cipher uses a separate monoalphabetic substitution cipher for each successive letter of plaintext, depending on a key.

The screenshot shows a C++ IDE with a file named `polyalphabetic.cpp`. The code implements the polyalphabetic substitution cipher algorithm. It includes headers for `stdio.h`, `conio.h`, and `string.h`. The `int main()` function prompts the user for a plaintext and a key, then calls the `encrypt` function to generate the ciphertext. The `encrypt` function uses a separate monoalphabetic substitution cipher for each successive letter of the plaintext, depending on the key.

```

1 #include <stdio.h>
2 #include <conio.h>
3 #include <string.h>
4 int main()
5 {
6     char pt[20], ct[20], key[20], rt[20];
7     printf("Enter the plain text:");
8     scanf("%s", pt);
9     printf("Enter the key:");
10    scanf("%s", key);
11    int i, j;
12    for (i = 0; i < strlen(pt); i++)
13    {
14        if (i == strlen(key))
15        {
16            j = 0;
17            key[i] = key[j];
18            j++;
19        }
20        for (j = 0; j < strlen(pt); j++)
21        {
22            ct[j] = ((pt[j] - 97) + (key[i] - 97) % 26) % 97;
23        }
24        printf("Enter the cipher text is: %s", ct);
25        for (i = 0; i < strlen(ct); i++)
26        {
27            if (ct[i] < key[i])
28            {
29                rt[i] = 26 + ((ct[i] - 97) - (key[i] - 97)) % 97;
30            }
31            else
32            {
33                rt[i] = ((ct[i] - 97) - (key[i] - 97) % 26) % 97;
34            }
35        }
36        printf("Enter the plain text is: %s", rt);
37        getch();
38    }
39 }

```

The execution output shows the plain text as "welcome", the key as "2", the cipher text as "HPWNZXP", and the plain text as "welcome".