

VARUN BHARGAVA - 241010282

DATA STRUCTURES TASK-8

Task 01: Josephus Problem:

(<https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab8/lab8-1.c>)

Write a program that simulates the Josephus Problem, where n players stand in a circle and every kth player are eliminated until only one player remains. The program should determine and print the position of the survivor (survivor).

Example:

Input: n = 7, k = 3

Output: The winner is player 4

The screenshot shows a terminal window with two panes. The left pane displays the source code for a C program named lab8-1.c. The right pane shows the terminal output.

Terminal Output:

```
lab8> C:\lab8-1.c > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node
5 {
6     int data;
7     struct Node *next;
8 };
9
10 struct Node *c1l(int n)
11 {
12     struct Node *head = NULL, *temp = NULL, *newNode;
13     for (int i = 1; i <= n; i++)
14     {
15         newNode = (struct Node *)malloc(sizeof(struct Node));
16         newNode->data = i;
17         if (head == NULL)
18         {
19             head = newNode;
20             head->next = head;
21             temp = head;
22         }
23         else
24         {
25             temp->next = newNode;
26             newNode->next = head;
27             temp = newNode;
28         }
29     }
30     return head;
31 }
32
33 int joseph(int n, int k)
34 {
35     struct Node *head = c1l(n);
36     struct Node *prev = NULL, *curr = head;
37
38     while (curr->next != curr)
39     {
40         for (int i = 1; i < k; i++)
41         {
42             prev = curr;
43             curr = curr->next;
44         }
45         prev->next = curr->next;
46         free(curr);
47         curr = prev->next;
48     }
49     int win = curr->data;
50     free(curr);
51     return win;
52 }
53
54 int main()
55 {
56     printf("n:");
57     int n;
58     scanf("%d", &n);
59     printf("k:");
60     int k;
61     scanf("%d", &k);
62     int win = joseph(n, k);
63     printf("winner: %d\n", win);
64     return 0;
65 }
```

```
● dsa\lab8> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab8\" ; if ($?) { gcc lab8-1.c -o lab8-1 }
if (?) { ./lab8-1 }
n:7
k:3
winner: 4
○ PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab8>
```

The terminal output shows the command to change directory, compile the program, run it, and finally print the result which is 4.

Task 02: Single Linked List – Longest Ascending Sequence:

(<https://github.com/varunnnb/dsa-sem3-iitnr/blob/main/lab8/lab8-2.c>)

Write a program using a single linked list that takes input from the user and prints the longest sequence of ascending order integers in it along with the length.

Example:

Input: 20, 2, 5, 17, 77, 7, 5, 3

Output: Longest ascending sequence: 2 5 17 77

The image shows two side-by-side screenshots of a code editor (VS Code) and a terminal window. The left screenshot displays the first version of the program, and the right one displays the second version. Both versions are implemented in C and use a linked list structure to find the longest ascending sequence.

Version 1 (Left):

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

struct Node
{
    int data;
    struct Node *next;
};

struct Node *createNode(int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void append(struct Node **head, int data)
{
    struct Node *newNode = createNode(data);
    if (*head == NULL)
    {
        *head = newNode;
        return;
    }
    struct Node *temp = *head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = newNode;
}

void longest(struct Node *head)
{
    if (head == NULL)
    {
        printf("list empty.\n");
        return;
    }
    struct Node *curr = head;
}
```

Version 2 (Right):

```
void longest(struct Node *head)
{
    if (head == NULL)
        ;
    struct Node *curr = head;
    struct Node *start = head;
    struct Node *bestStart = head;

    int length = 1, bestLength = 1;

    while (curr->next != NULL)
    {
        if (curr->next->data > curr->data)
        {
            length++;
        }
        else
        {
            if (length > bestLength)
            {
                bestLength = length;
                bestStart = start;
            }
            start = curr->next;
            length = 1;
        }
        curr = curr->next;
    }

    if (length > bestLength)
    {
        bestLength = length;
        bestStart = start;
    }
}

printf("longest seq: ");
curr = bestStart;
for (int i = 0; i < bestLength; i++)
{
    printf("%d ", curr->data);
    curr = curr->next;
}
```

The terminal window in both screenshots shows the execution of the program. It starts by navigating to the directory containing the source code, then compiles the program using gcc. It prompts the user to enter the number of elements (6), followed by the integers 12, 14, 31, 43, 12, and 17. The program then outputs the longest sequence found: 12 14 31 43, with a length of 4.

```

lab8 > C lab8-2.c > createNode(int)
34 void longest(struct Node *head)
75     for (int i = 0; i < bestLength; i++)
76     {
77         printf("%d ", curr->data);
78         curr = curr->next;
79     }
80     printf("\nLength = %d\n", bestLength);
81 }
82
83 int main()
84 {
85     struct Node *head = NULL;
86     int n, value;
87
88     printf("Enter number of elements: ");
89     scanf("%d", &n);
90
91     printf("Enter %d integers:\n", n);
92     for (int i = 0; i < n; i++)
93     {
94         scanf("%d", &value);
95         append(&head, value);
96     }
97
98     longest(head);
99
100    return 0;
101 }
102

```

PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab8> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab8-2" ; if (\$?) { gcc lab8-2.c -o lab8-2 }
if (\$?) { .\lab8-2 }
Enter number of elements: 6
Enter 6 integers:
12
14
31
43
12
17
longest seq: 12 14 31 43
Length = 4
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab8> []

Task 03: Anagram Check:

(<https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab8/lab8-3.c>)

Check whether two input strings are anagrams or not

Example:

Input: listen, silent

Output: Strings are Anagrams

```

C lab8-1.c U C lab8-2.c U C lab8-3.c U
1 #include <stdio.h>
2 #include <string.h>
3
4 int areAnagrams(char str1[], char str2[])
5 {
6     int count[256] = {0};
7
8     if (strlen(str1) != strlen(str2))
9         return 0;
10
11    for (int i = 0; str1[i] && str2[i]; i++)
12    {
13        count[(unsigned char)str1[i]]++;
14        count[(unsigned char)str2[i]]--;
15    }
16
17    for (int i = 0; i < 256; i++)
18    {
19        if (count[i] != 0)
20            return 0;
21    }
22    return 1;
23 }
24
25 int main()
26 {
27     char str1[100], str2[100];
28
29     printf("Enter first string: ");
30     scanf("%s", str1);
31     printf("Enter second string: ");
32     scanf("%s", str2);
33
34     if (areAnagrams(str1, str2))
35         printf("Strings are Anagrams\n");
36     else
37         printf("Strings are Not Anagrams\n");
38
39     return 0;
40 }
41

```

PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab8-3" ; if (\$?) { gcc lab8-3.c -o lab8-3 } ; if (\$?) { .\lab8-3 }
Enter first string: varun
Enter second string: runva
Strings are Anagrams
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa> []