DAA Lab-9

11/03/25 – 230962326– Davasam Karthikeya – AIMLB

Solved question (Countsort):

```
#include <stdio.h>

void countSort(int arr[], int n) {
    int sorted[32], count[32] = {};

    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if(arr[i] < arr[j])
                count[j]++;
            else
                count[i]++;
        }
    }

    for (int i = 0; i < n; i++)
        sorted[count[i]] = arr[i];

    printf("Sorted: ");
    for (int i = 0; i < n; i++)
        printf("%d ", sorted[i]);
}

void main() {
    int n, arr[32] = {};

    printf("Enter array size: ");
    scanf("%d", &n);

    printf("Enter nums:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    countSort(arr, n);
    printf("\n");
}
```

Output

```
Enter array size: 6
Enter nums:
62
31
84
96
19
47
Sorted: 19 31 47 62 84 96
```

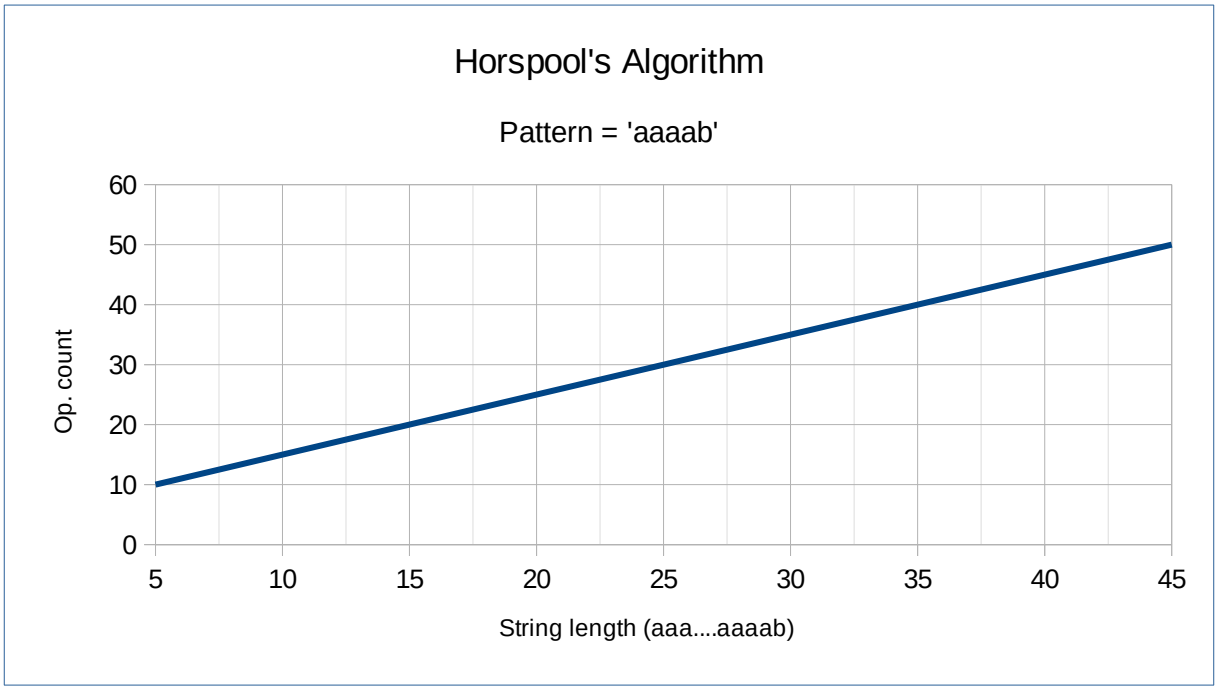
Question 1 (Horspool's algorithm):

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

#define SIZE 256
int table[SIZE];
```

```
void shifter(char pat[]) {
    int len = strlen(pat);

    for(int i = 0; i < SIZE; i++)
        table[i] = len;

    for(int i = 0; i < len - 1; i++)
        table[pat[i]] = len - i - 1;
}

int horspool(char str[],char pat[]) {
    int k, strLen = strlen(str), patLen = strlen(pat);

    for(int i = patLen - 1; i < strLen; i += table[str[i]]) {
        for(k = 0; k < patLen && pat[patLen - k - 1] == str[i - k]; k++);

        if(k == patLen)
            return i - patLen + 1;
    }

    return -1;
}

void main() {
    char str[64], pat[64];

    printf("Enter the str: ");
    gets(str);

    printf("Enter the pattern: ");
    gets(pat);

    shifter(pat);

    int pos = horspool(str, pat);

    if(pos < 0)
        printf("Not found\n");
    else
        printf("Position: %d\n", pos);
}
```

Output

Enter the str: Car search
Enter the pattern: ear
Position: 5



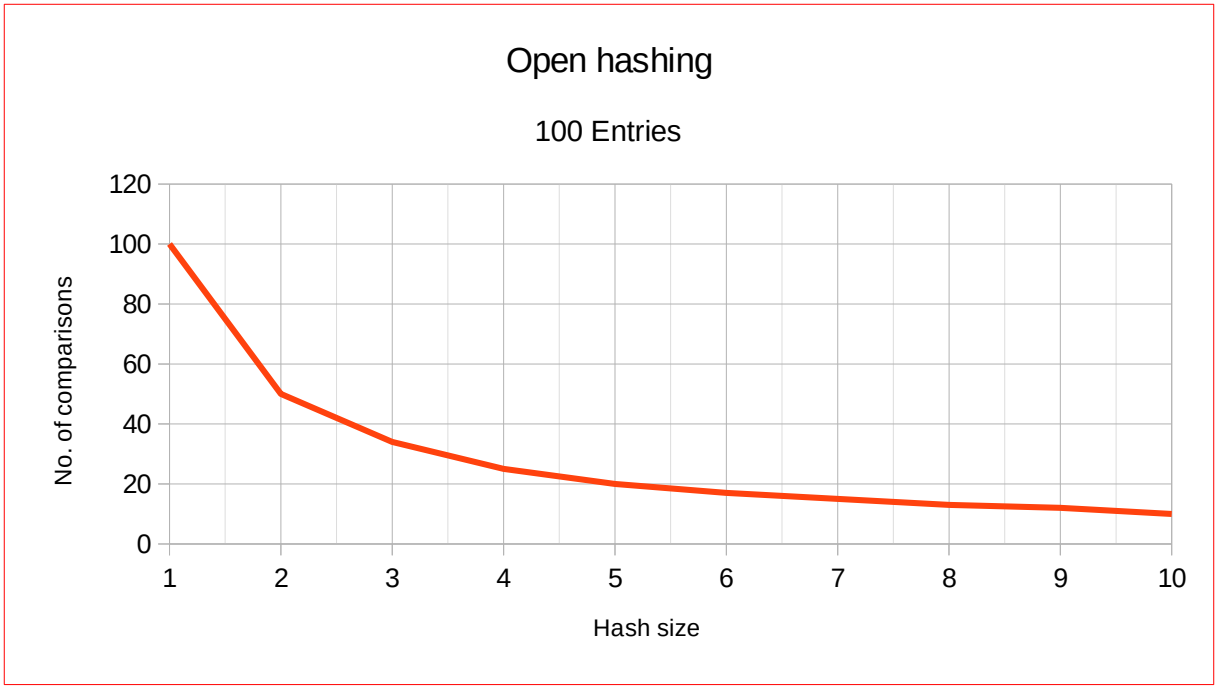
Question 2 (Open hashing):

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

#define LF 5

typedef struct node {
    int data;
    struct node *next;
} NODE;

void insert(NODE *root[]) {
    int ind, val;

    printf("Enter value: ");
    scanf("%d", &val);
    ind = val % LF;

    NODE *newNode = malloc(sizeof(NODE)), *temp;
    newNode -> data = val;
    newNode -> next = NULL;

    if (!root[ind])
        root[ind] = newNode;
    else {
        for(temp = root[ind]; temp -> next; temp = temp -> next);

        temp->next = newNode;
    }
}

void search(NODE *root[]) {
    int key, ind;
    NODE *temp;

    printf("Enter key: \n");
    scanf("%d", &key);
    ind = key % LF;
```

```
    for(temp = root[ind]; temp -> data != key
    && temp -> next; temp = temp -> next);

    if(temp -> data == key) {
        printf("Index: %d\n", ind);
        return;
    }

    printf("Not found\n");
}

void main() {
    NODE *hash[LF] = {};
    int ch;

    printf("Options:\n0. Exit\n1. Insert\n2. Display\n3. Search");
    do {
        printf("\nEnter option: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            insert(hash);
            break;
        case 2:
            search(hash);
            break;
        }
    } while (ch);
}
```

Output

```
Options:
0. Exit
1. Insert
2. Search
Enter option: 1
Enter value: 19

Enter option: 2
Enter key:
19
Index: 4

Enter option: 0
```



Question 3 (Closed hashing):

```
#include <stdio.h>

#define SIZE 7

void insert(int hash[]) {
    int key, ind;

    printf("Enter value: ");
    scanf("%d", &key);
    ind = key % SIZE;

    for (int i = 0; i < SIZE; i++) {
        ind = (ind + 1) % SIZE;
        if (!hash[ind]) {
            hash[ind] = key;
            return;
        }
    }

    printf("Insertion failed\n");
}

void search(int hash[]) {
    int key, ind;

    printf("Enter key: \n");
    scanf("%d", &key);
    ind = key % SIZE;

    for (int i = 0; i < SIZE; i++)
    {
        ind = (ind + 1) % SIZE;
        if (hash[ind] == key)
        {
            printf("Index: %d\n", ind);
            return;
        }
```

```
    }

    printf("Not found\n");
}

void main() {
    int ch, hash[SIZE] = {};

    printf("Options:\n0. Exit\n1. Insert\n2. Search");
    do {
        printf("\nEnter option: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            insert(hash);
            break;
        case 2:
            search(hash);
            break;
        }
    } while (ch);
}
```

Output

```
Options:
0. Exit
1. Insert
2. Search
Enter option: 1
Enter value: 19

Enter option: 2
Enter key:
19
Index: 6

Enter option: 0
```