**Assignment: 12**

**EXP 1. WAP to check for the smallest and largest subarrays in an array whose sum is 15. Generate the array randomly. Implement hashing and linear probing to avoid collision in a given set of keys.**

**Coding:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define MAX\_SIZE 10

#define SUM\_VALUE 15

int arr[MAX\_SIZE], table[MAX\_SIZE];

int random\_num()

{

    return (rand()%20);

}

void initializeTable()

{

    for(int i=0; i<MAX\_SIZE; i++)

        table[i] = -1;

}

int hashFn(int key)

{

    return key % MAX\_SIZE;

}

int linearProbe(int key)

{

    int i = 0;

    while(1)

    {

        int index = (hashFn(key) + i) % MAX\_SIZE;

        if(table[index] == -1)

            return index;

        i++;

    }

}

void insert(int key)

{

    int index = hashFn(key);

    if(table[index] == -1)

        table[index] = key;

    else

        table[linearProbe(key)] = key;

}

void displayArray(int arr[], int n)

{

    for(int i=0; i<n; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

void smallestSubarray()

{

    int min\_len = MAX\_SIZE;

    int start = 0, end = 0;

    int sum = 0;

    for(int i=0; i<MAX\_SIZE; i++)

        sum += arr[i];

    if(sum == SUM\_VALUE)

    {

        min\_len = MAX\_SIZE;

        start = 0;

        end = MAX\_SIZE - 1;

    }

    else

    {

        sum = 0;

        int i = 0;

        while(i<MAX\_SIZE)

        {

            sum += arr[i];

            if(sum > SUM\_VALUE)

            {

                int j = i-1;

                while(j>=0 && sum>SUM\_VALUE)

                {

                    sum -= arr[j];

                    j--;

                }

                int len = i - j;

                if(len < min\_len)

                {

                    min\_len = len;

                    start = j + 1;

                    end = i;

                }

                i++;

            }

            else if(sum == SUM\_VALUE)

            {

                int len = i + 1;

                if(len < min\_len)

                {

                    min\_len = len;

                    start = 0;

                    end = i;

                }

                i++;

            }

            else

                i++;

        }

    }

    printf("The smallest subarray is: ");

    displayArray(arr+start, end-start+1);

    printf("Length of the smallest subarray is: %d\n", min\_len);

}

void largestSubarray()

{

    int max\_len = 0;

    int start = 0, end = 0;

    int sum = 0;

    for(int i=0; i<MAX\_SIZE; i++)

        sum += arr[i];

    if(sum == SUM\_VALUE)

    {

        max\_len = MAX\_SIZE;

        start = 0;

        end = MAX\_SIZE - 1;

    }

    else

    {

        sum = 0;

        int i = 0;

        while(i<MAX\_SIZE)

        {

            sum += arr[i];

            if(sum > SUM\_VALUE)

            {

                int j = i-1;

                while(j>=0 && sum>SUM\_VALUE)

                {

                    sum -= arr[j];

                    j--;

                }

                int len = i - j;

                if(len > max\_len)

                {

                    max\_len = len;

                    start = j + 1;

                    end = i;

                }

                i++;

            }

            else if(sum == SUM\_VALUE)

            {

                int len = i + 1;

                if(len > max\_len)

                {

                    max\_len = len;

                    start = 0;

                    end = i;

                }

                i++;

            }

            else

                i++;

        }

    }

    printf("The largest subarray is: ");

    displayArray(arr+start, end-start+1);

    printf("Length of the largest subarray is: %d\n", max\_len);

}

int main()

{

    srand(time(0));

    printf("The array is: ");

    for(int i=0; i<MAX\_SIZE; i++)

    {

        arr[i] = random\_num();

        printf("%d ", arr[i]);

    }

    printf("\n");

    initializeTable();

    for(int i=0; i<MAX\_SIZE; i++)

        insert(arr[i]);

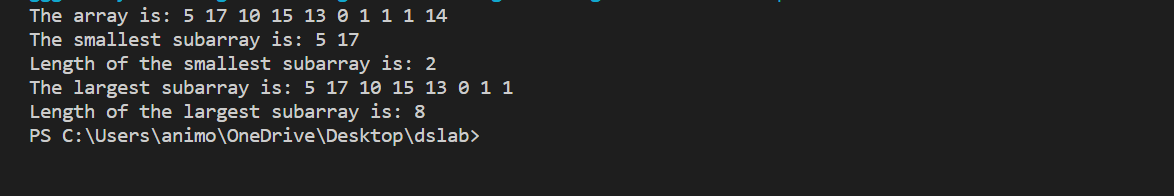
    smallestSubarray();

    largestSubarray();

    return 0;

}

**Output:**

****

**Exp.2. WAP to implement the functionality of double hashing.**

**Coding:**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define TABLE\_SIZE 13

struct hash{

    int value;

    int flag;

};

struct hash table[TABLE\_SIZE];

int hash1(int value){

    return (value % TABLE\_SIZE);

}

int hash2(int value){

    return 7 - (value % 7);

}

void insert(int value){

    int h1 = hash1(value);

    int h2 = hash2(value);

    if (table[h1].flag == 0){

        table[h1].value = value;

        table[h1].flag = 1;

    }

    else{

        int i;

        int loc;

        int h;

        for (i = 1; i < TABLE\_SIZE; i++){

            loc = (h1 +  i \* h2) % TABLE\_SIZE;

            if (table[loc].flag == 0){

               table[loc].flag = 1;

               table[loc].value = value;

               break;

            }

        }

        if (i == TABLE\_SIZE){

            printf("\nHash table full !! \n");

            return;

        }

    }

    printf("\n%d is inserted !! \n", table[h1].value);

}

int search(int value){

    int h1 = hash1(value);

    int h2 = hash2(value);

    if (table[h1].flag == 1 && table[h1].value == value){

        printf("\nElement %d found !! \n",value);

        return h1;

    }

    int i;

    int loc;

    for (i = 0; i < TABLE\_SIZE; i++){

        loc = (h1 +  i \* h2 ) % TABLE\_SIZE;

        if (table[loc].flag == 1 && table[loc].value == value){

            printf("\nElement %d found !! \n",value);

            return loc;

        }

    }

    return -1;

}

void display(){

    printf("\nHash Table: \n");

    int i;

    for (i = 0; i < TABLE\_SIZE; i++){

        printf("table[%d] = %d \n",i,table[i].value);

    }

}

int main(){

    int value,choice,i;

    srand(time(0));

    for (i = 0; i < TABLE\_SIZE; i++){

        table[i].flag = 0;

        table[i].value = 0;

    }

    while(1){

        printf("\n1. Insert a value \n");

        printf("2. Display table \n");

        printf("3. Search a value \n");

        printf("4. Exit \n");

        printf("Enter your choice \n");

        scanf("%d",&choice);

        switch(choice){

            case 1:

                printf("\nEnter value to insert in Hash Table \n");

                scanf("%d",&value);

                insert(value);

                break;

            case 2:

                display();

                break;

            case 3:

                printf("\nEnter value to be searched \n");

                scanf("%d",&value);

                i = search(value);

                break;

            case 4:

                printf("\nExitting !! \n");

                exit(0);

            default:

                printf("\nWrong Choice \n");

                break;

        }

    }

    return 0;

}

**Output:**

