In-Lab – 1(page No: 185)

```
#include<stdio.h>
#define size 100
int table[size],m,n;
void insert()
{
   int key,index;
   scanf("%d",&key);
   index=key%m;
   if(table[index] == -1)
   {
      table[index]=key;
      printf("%d->%d\n", key,index);
   }
   else
   {
      printf("%d->Collsion Occured\n",key);
   }
}
int main()
{
   int i;
   scanf("%d",&n);
   scanf("%d",&m);
   for(i=0;i<m;i++)
      table[i]=-1;
   for(i=0;i<n;i++)
      insert();
   return 0;
}
```

In-Lab – 2(page No: 187)

```
#include <stdio.h>
int main()
{
   int tc;
   scanf("%d", &tc);
   while (tc--)
   {
      int n,i,j,count;
      scanf("%d", &n);
      int arr[n];
```

```
        for (i = 0; i < n; i++)
        {
            scanf("%d", &arr[i]);
        }
        for (i = 0; i < n; i++)
        {
            count = 0;
            for (j = 0; j < n; j++)
            {
                if(arr[i] == arr[j])
                    count++;
            }
            printf("%d ", count);
        }
        printf("\n");
    }
    return 0;
}
```

In-Lab – 3(page No: 189)

```
#include<stdio.h>
#define size 100
int table[size],m,n;
void insert()
{
    int key,index,i;
    scanf("%d",&key);
    for(i=0;i<n;i++)
    {
        index=(key+i)%m;
        if(table[index] == -1)
        {
            table[index]=key;
            printf("%d->%d\n",key,index);
            break;
        }
    }
}
int main()
{
    int i;
    scanf("%d",&n);
    scanf("%d",&m);
```

```
    for(i=0;i<m;i++)
        table[i]=-1;
    for(i=0;i<n;i++)
        insert();
    return 0;
}
```

Post-Lab – 1(page No: 191)

```
int cmp(const void *a, const void *b)
{
    return (*(int*)a - *(int*)b);
}

bool containsDuplicate(int* nums, int numsSize)
{
    qsort(nums, numsSize, sizeof(int), cmp);

    for (int i = 1; i < numsSize; i++)
        if (nums[i] == nums[i-1])
            return true;

    return false;
}
```

Post-Lab – 2(page No:193)

```
bool isAnagram(char* s, char* t)
{
    if (strlen(s) != strlen(t))
    {
        return false;
    }
    int len = strlen(s);
    int* charCountS = (int*)calloc(26, sizeof(int));
    int* charCountT = (int*)calloc(26, sizeof(int));
    for (int i = 0; i < len; i++)
    {
        charCountS[s[i] - 'a']++;
    }
    for (int i = 0; i < len; i++)
    {
        charCountT[t[i] - 'a']++;
```

```
    }
    bool result = true;
    for (int i = 0; i < 26; i++)
    {
        if (charCountS[i] != charCountT[i])
        {
            result = false;
            break;
        }
    }
    free(charCountS);
    free(charCountT);
    return result;
}
```

Skill Lab-1(page No:195)

```
char* stringHash(char* s, int k)
{
    int n = strlen(s);
    int length = n / k;
    char *result = (char*)malloc(length + 1);
    if (!result)
    {
        return NULL;
    }
    int index = 0,i,j,sum;
    for (i = 0; i < n; i += k)
    {
        sum = 0;
        for (j = i; j < i + k; j++)
        {
            sum += s[j] - 'a';
        }
        result[index++] = (sum % 26) + 'a';
    }
    result[index] = '\0';
    return result;
}
```

Skill Lab-2(page No:197)

```
bool digitCount(char* num)
{
   int n = strlen(num);
   int table[10] = {0};
   int i;
   for (i = 0; i < n; i++)
   {
      table[num[i] - '0']++;
   }
   for (i = 0; i < n; i++)
   {
      if (table[i] != (num[i] - '0'))
      {
         return false;
      }
   }
   return true;
}
```

Skill Lab-3(page No:199)

```
#include <stdio.h>
int main()
{
   int n;
   scanf("%d", &n);
   int a[n];
   for (int i = 0; i < n; i++)
   {
      scanf("%d", &a[i]);
   }
   int res = 0,i,j;
   for (i = 0; i < n; i++) {
      for (j = i + 1; j < n; j++) {
         if (a[i] == a[j] * a[j]) {
            res += 1;
         }
    }
}
   printf("%d", res);
   return 0;
}
```

Skill Lab-4(page No:201)

```
#include <stdio.h>
const int M = 999983;

int f(int x)
{
   return x % M;
}
int main()
{
   for(int i = 0; i < 5; i++)
   {
      int x;
      scanf("%d", &x);
      printf("x = %d, f(x) = %d\n", x, f(x));
   }
   return 0;
}
```

Skill Lab-5(page No:203)

```
int* findDuplicates(int* nums, int numsSize, int* returnSize)
{
   *returnSize = 0;
   int *result = (int*)malloc(numsSize * sizeof(int));
   int i,index;
   for (i = 0; i < numsSize; i++)
   {
      index = abs(nums[i]) - 1;
      if (nums[index] < 0)
      {
         result[(*returnSize)++] = index + 1;
      }
      else
      {
         nums[index] = -nums[index];
      }
   }
   return result;
}
```