```
//Calculate Nth term

int term, t1 = a, t2 = b, t3 = c;

if (n == 1)

term = t1;

else if (n == 2)

 term = t2;

else if (n == 3)

term = t3;

else {
 for (int i = 4; i <= n; i++) {

term = t1 + t2 + t3; t1 = t2;
 t2 = t3;
 t3 = term;

} }

return term;

}

//Students Marks Sum

int sum = 0;
 for(int i = (gender == 'b' ? 0 : gender == 'g' ? 1 : -1); i < number_of_students;

i+=2) {
 sum += marks[i];

}

return sum;

// Function to calculate the Nth Tribonacci number

int tribonacci(int n) {

if (n == 0) return 0;
 if (n == 1 || n == 2) return 1; int a = 0, b = 1, c = 1, next; for (int i = 3; i <= n; i++) {

next = a + b + c; // Calculate the next term

a = b; // Update a to the next term
 b = c; // Update b to the next term
 c = next; // Update c to the next term

}

return c;

}
```

Module 4

```
 // 1 SORTING ARRAY OF STRINGS
```

```
int sort_by_length(const char* a, const char* b){
    if(strlen(a) != strlen(b))
        return strlen(a) > strlen(b);
    else
        return strcmp(a, b) >  0;
}
//2  1D ARRAYS IN C
int main()
{
    int n;
    scanf("%d", &n);
    // Create a dynamic array of size n
    int* arr = (int*)malloc(n * sizeof(int));
    // Read the values from stdin and store them in the array
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    // Calculate the sum of all elements in the array
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

//3 Array Reversal
for (i = 0; i < n; i++)
scanf("%d", &arr[i]);
for (i = n - 1; i >= 0; i--)
printf("%d ", arr[i]);


//4   Binary Search Tree: Insertion
struct node* insert(struct node* root, int data) {
```

```
    if (root == NULL) {
        struct node* newNode = (struct node*)malloc(sizeof(struct node));
        newNode->data = data;
        newNode->left = NULL;
        newNode->right = NULL;
        return newNode;
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
    } else {
        root->right = insert(root->right, data);
    }

    return root;
}
//5 Remove Duplicates from Sorted Array
int main() {
// Example input
int nums[] = {1, 1, 2, 2, 3, 3, 4};
int i;
int numsSize = sizeof(nums) / sizeof(nums[0]);
// Calling removeDuplicates function
int newSize = removeDuplicates(nums, numsSize);



Module 5
//1 Permutation Of Strings
char *tmp = s[k];
s[k] = s[l];
s[l] = tmp;
i = k+1, j = n-1;
while (i < j) {
```

```
tmp = s[i];
s[i++] = s[j];
s[j--] = tmp;
}
return 1;
}


//2 2D Array
int main()
{
int i,j,k;
int arr[6][6],temp=-9999,a,b;
for(i=0;i<6;i++)
for(j=0;j<6;j++)
scanf("%d",&arr[i][j]);
for(i=0;i<=3;i++)
for(j=0;j<=3;j++)
{
a = arr[i][j]+arr[i][j+1]+arr[i][j+2]+arr[i+1][j+1]+arr[i+2][j]+arr[i+2][j+1]+arr[i+2][j+2];
if(temp < a)
temp = a ;
}
return 0;
}


//3 Dynamic
if (query_type == 1) {
// Add a book with y pages to shelf idx
shelves[idx] = (int*)realloc(shelves[idx], (sizes[idx] + 1) * sizeof(int));
shelves[idx][sizes[idx]] = y; // Add the number of pages
sizes[idx]++; // Increment the count of books on shelf idx
```

```
} else if (query_type == 2) {
// Retrieve the number of pages in the y-th book on shelf idx
last_ans = shelves[idx][y % sizes[idx]];
printf("%d\n", last_ans);
} else if (query_type == 3) {
// Print the total number of books on shelf idx
printf("%d\n", sizes[idx]);
}

//4. Printing Tokens
for(i=0;i<strlen(s);i++){
if(*(s+i)==' ')
printf("\n");
Else
printf("%c",*(s+i));
}free(s);
Return 0;
}

//5. Index of first occurrence of a string
while (haystack[i]!='\0' && needle[j]!='\0' ) {
if (haystack[i] == needle[j]) {
i++; j++;
}
else {
i++; j = 0;
}
}
if (j == nsize)
res =(i- nsize);
else
res=-1;
return  res;}
```