**Technical Proficiency Enhancement Course -1 BITTP308 Partial Codes**

**Regular Programs**

1. **Sum and difference of two numbers**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

// code

return 0;

}

**2. Playing with characters**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

//code

return 0;

}

**3.Conditional statements in C**

#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

int main()

{

char\* n\_endptr;

char\* n\_str = readline();

int n = strtol(n\_str, &n\_endptr, 10);

if (n\_endptr == n\_str || \*n\_endptr != '\0') { exit(EXIT\_FAILURE); }

//code

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char data[1024];

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) { break; }

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') { break; }

size\_t new\_length = alloc\_length << 1;

if (!data) { break; }

alloc\_length = new\_length;

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

}

return data;

}

**4. Bitwise Operators**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

void calculate\_the\_maximum(int n, int k) {

//code

}

int main() {

    int n, k;

    scanf("%d %d", &n, &k);

    calculate\_the\_maximum(n, k);

    return 0;

}

1. **Printing Patterns Using Loops**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

int n;

scanf("%d", &n);

//code

}

1. **Correctness and loop invariants**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include <assert.h>

void insertionSort(int N, int arr[]) {

//code

}

int main(void) {

int N;

scanf("%d", &N);

int arr[N], i;

for(i = 0; i < N; i++) {

scanf("%d", &arr[i]);

}

insertionSort(N, arr);

return 0;

}

1. **Triangle Numbers**

#include <stdio.h>

// Function to calculate the nth triangular number

//code

int main() {

int n, i;

// Input for how many triangular numbers to generate

printf("Enter the number of triangular numbers to generate: ");

scanf("%d", &n);

// Generate and display the first n triangular numbers

printf("The first %d triangular numbers are:\n", n);

for (i = 1; i <= n; i++) {

printf("%d ", triangularNumber(i));

}

printf("\n");

return 0;

}

1. **Small And Large Triangle**

#include <stdio.h>

// Function to calculate the area of a triangle

//code

int main() {

float base1, height1, base2, height2, area1, area2;

// Input for the first triangle

printf("Enter the base and height of the first triangle: ");

scanf("%f %f", &base1, &height1);

// Input for the second triangle

printf("Enter the base and height of the second triangle: ");

scanf("%f %f", &base2, &height2);

// Calculate the areas of both triangles

area1 = calculateArea(base1, height1);

area2 = calculateArea(base2, height2);

// Compare the areas and determine which triangle is larger

printf("\nArea of first triangle: %.2f", area1);

printf("\nArea of second triangle: %.2f", area2);

//code

return 0;

}

1. **For Loop in C**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

int a, b,i;

scanf("%d\n%d", &a, &b);

// Complete the code.

return 0;

}

1. **Calculate the Nth term**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

//Complete the following function.

int find\_nth\_term(int n, int a, int b, int c) {

//code

}

int main() {

int n, a, b, c;

scanf("%d %d %d %d", &n, &a, &b, &c);

int ans = find\_nth\_term(n, a, b, c);

printf("%d", ans);

return 0;

}

1. **Students Marks Sum**

#include<stdio.h>

int main() {

int n, i;

float marks, sum = 0;

// Input the number of students

printf("Enter the number of students: ");

scanf("%d", &n);

//code

// Display the total sum of marks

printf("\nThe sum of marks for %d students is: %.2f\n", n, sum);

return 0;

}

1. **Sum of Digits of a five Digit Number**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

int n;

scanf("%d", &n);

//Code

}

1. **Sorting Array Of Strings**

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

int lexicographic\_sort(const char\* a, const char\* b){

return strcmp(a, b) > 0;

}

int lexicographic\_sort\_reverse(const char\* a, const char\* b){

return strcmp(a, b) <= 0;

}

int sort\_by\_number\_of\_distinct\_characters(const char\* a, const char\* b){

int c1 = 0, c2 = 0;

int hsh1[26] = {0}, hsh2[26] = {0};

int n1 = strlen(a);

int n2 = strlen(b);

int i;

for(i = 0; i < n1; i++){

hsh1[a[i] - 'a'] = 1;

}

for(i = 0; i < n2; i++){

hsh2[b[i] - 'a'] = 1;

}

for( i = 0; i < 26; i++){

if(hsh1[i])

c1++;

if(hsh2[i])

c2++;

}

if( c1 != c2)

return c1 > c2;

else

return strcmp(a, b) > 0;

}

int sort\_by\_length(const char\* a, const char\* b){

//code

}

void string\_sort(char\*\* arr,const int len,int (\*cmp\_func)(const char\* a, const char\* b))

{ int i;

for( i = 1; i < len; i++){

int j = i;

char\* p = arr[i];

while(j > 0){

if((\*cmp\_func)(arr[j-1],p) > 0 )

arr[j] = arr[j-1];

else

break;

j--;

}

arr[j] = p;

}

}

int main()

{

int n,i;

scanf("%d", &n);

char\*\* arr;

arr = (char\*\*)malloc(n \* sizeof(char\*));

for(i = 0; i < n; i++){

\*(arr + i) = malloc(1024 \* sizeof(char));

scanf("%s", \*(arr + i));

\*(arr + i) = realloc(\*(arr + i), strlen(\*(arr + i)) + 1);

}

string\_sort(arr, n, lexicographic\_sort);

for(i = 0; i < n; i++)

printf("%s\n", arr[i]);

printf("\n");

string\_sort(arr, n, lexicographic\_sort\_reverse);

for(i = 0; i < n; i++)

printf("%s\n", arr[i]);

printf("\n");

string\_sort(arr, n, sort\_by\_length);

for(i = 0; i < n; i++)

printf("%s\n", arr[i]);

printf("\n");

string\_sort(arr, n, sort\_by\_number\_of\_distinct\_characters);

for(i = 0; i < n; i++)

printf("%s\n", arr[i]);

printf("\n");

}

**14. 1D ARRAYS IN C**

#include <stdio.h>

#include <stdlib.h>

int main()

{

//code

printf("%d\n", sum);

free(arr);

return 0;

}

**15.** **Array Reversal**

#include <stdio.h>

#include <stdlib.h>

int main() {

int n, arr[1000], i;

scanf("%d", &n);

//code

printf("\n");

return 0;

}

**16. Binary Search Tree: Insertion**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

struct node {

int data;

struct node \*left;

struct node \*right;

};

void preOrder( struct node \*root) {

if( root == NULL )

return;

printf("%d ",root->data);

preOrder(root->left);

preOrder(root->right);

}

struct node\* insert(struct node\* root, int data) {

//code

}

int main() {

struct node\* root = NULL;

int t;

int data;

scanf("%d", &t);

while(t-- > 0) {

scanf("%d", &data);

root = insert(root, data);

}

preOrder(root);

return 0;

}

**17. Permutation Of Strings**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int next\_permutation(int n, char \*\*s)

{

int i,j;

int k = -1;

for ( i = 0; i < n-1; i++) {

if (strcmp(s[i], s[i+1]) < 0)

k = i;

} if ( k== -1) return 0;

int l = -1;

for ( i = k+1; i < n; i++) {

if (strcmp(s[k], s[i]) < 0)

l = i;

}

//code

}

int main()

{

char \*\*s;

int n,i;

scanf("%d", &n);

s = calloc(n, sizeof(char\*));

for ( i = 0; i < n; i++)

{

s[i] = calloc(11, sizeof(char));

scanf("%s", s[i]);

}

do

{

for ( i = 0; i < n; i++)

printf("%s%c", s[i], i == n - 1 ? '\n' : ' ');

} while (next\_permutation(n, s));

for ( i = 0; i < n; i++)

free(s[i]);

free(s);

return 0;

}

**18. 2D Array**

#include <assert.h>

#include <ctype.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\* ltrim(char\*);

char\* rtrim(char\*);

char\*\* split\_string(char\*);

int parse\_int(char\*);

//code

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length -data\_length, stdin);

if (!line) {

break;

}

data\_length += strlen(cursor);

if (data\_length < alloc\_length -1 || data[data\_length -1] == '\n') {

break;

}

alloc\_length <<= 1;

data = realloc(data, alloc\_length);

if (!data) {

data = '\0';

break;

}

} if (

data[data\_length -

1] == '\n') { data[data\_length -1] = '\0';

data = realloc(data, data\_length);

if (!data) {

data = '\0';

}

} else {

data = realloc(data, data\_length + 1);

if (!data) {

data = '\0';

} else {

data[data\_length] = '\0';

}

} return data;

}

char\* ltrim(char\* str) {

if (!str) {

return '\0';

} if (!\*str) {

return str;

}

while (\*str != '\0' && isspace(\*str)) {

str++;

} return str;

}

char\* rtrim(char\* str) {

if (!str) {

return '\0';

} if (!\*str) {

return str;

}

char\* end = str + strlen(str) -1;

while (end >= str && isspace(\*end)) {

end--;

}

\*(end + 1) = '\0';

return str;

}

char\*\* split\_string(char\* str) {

char\*\* splits = NULL;

char\* token = strtok(str, " ");

int spaces = 0;

while (token) {

splits = realloc(splits, sizeof(char\*) \* ++spaces);

if (!splits) {

return splits;

}

splits[spaces -1] = token;

token = strtok(NULL, " ");

} return splits;

}

int parse\_int(char\* str) {

char\* endptr;

int value = strtol(str, &endptr, 10);

if (endptr == str || \*endptr != '\0') {

exit(EXIT\_FAILURE);

} return value;

}

**19. Dynamic array**

#include <stdio.h>

#include <stdlib.h>

int main() {

int n, q,i=0;

scanf("%d %d", &n, &q);

// Create an array of dynamic arrays for the shelves

int\*\* shelves = (int\*\*)malloc(n \* sizeof(int\*));

int\* sizes = (int\*)malloc(n \* sizeof(int)); // To keep track of the number of books in each shelf

int last\_ans = 0;

// Initialize sizes

for ( i = 0; i < n; i++) {

sizes[i] = 0;

shelves[i] = NULL; // Initialize each shelf to NULL

}

// Process each query

for ( i = 0; i < q; i++) {

int query\_type, x, y;

scanf("%d %d %d", &query\_type, &x, &y);

// Calculate the index for the shelf

int idx = (x ^ last\_ans) % n;

//code

}

// Free allocated memory

for ( i = 0; i < n; i++) {

free(shelves[i]); // Free each shelf

} free(shelves); // Free the shelves array

free(sizes); // Free the sizes array

return 0;

}

**20. Printing Tokens**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

char \*s;

int i;

s = malloc(1024 \* sizeof(char));

scanf(“%[^\n]”, s);

s = realloc(s, strlen(s) + 1);

//code

return 0;

}

Additional Programs

1. **Longest Substring Without Repeating Characters**

#include <stdio.h>

#include <string.h>

int lengthOfLongestSubstring(char \*s) {

//code

}

int maxLength = 0;

int start = 0; // Left pointer of the window

for (int end = 0; end < n; end++) {

char currentChar = s[end];

// If the character has been seen before, move the start pointer

// We use max() to ensure that start only moves forward

if (lastIndex[(int)currentChar] >= start) {

start = lastIndex[(int)currentChar] + 1;

}

// Update the last occurrence of the current character

lastIndex[(int)currentChar] = end;

// Calculate the length of the current valid substring

int currentLength = end - start + 1;

if (currentLength > maxLength) {

maxLength = currentLength;

}

}

return maxLength;

}

int main() {

char s1[] = "abcabcbb";

printf("Length of longest substring without repeating characters in '%s': %d\n", s1, lengthOfLongestSubstring(s1));

char s2[] = "bbbbb";

printf("Length of longest substring without repeating characters in '%s': %d\n", s2, lengthOfLongestSubstring(s2));

char s3[] = "pwwkew";

printf("Length of longest substring without repeating characters in '%s': %d\n", s3, lengthOfLongestSubstring(s3));

return 0;

}

1. **Happy Number**

#include<stdio.h>

//code

int main()

{

int num,sum=0;

printf("Enter the number");

scanf("%d",&num);

//code

if(num==1){

printf("Happy Number");

}

else{

printf("not Happy Number");

}

return 0;

}

1. **Nth Tribonacci number**

#include <stdio.h>

// Function to calculate the Nth Tribonacci number

//code

int main() {

int n;

// Input the value of N

printf("Enter the value of N: ");

scanf("%d", &n);

// Calculate and display the Nth Tribonacci number

printf("The %dth Tribonacci number is: %d\n", n, tribonacci(n));

return 0;

}

1. **Remove Duplicates from Sorted Array**

#include <stdio.h>

// Function to remove duplicates from a sorted array

int removeDuplicates(int\* nums, int numsSize) {

// Edge case: if the array is empty, no unique elements exist

if (numsSize == 0) {

return 0;

}

// k will track the index of the last unique element

int k = 1;

int i;

// Start from the second element (index 1)

for (i = 1; i < numsSize; i++) {

// If the current element is different from the previous one, it's unique

if (nums[i] != nums[i - 1]) {

// Place the unique element at position k

nums[k] = nums[i];

k++; // Increment k to track the number of unique elements

}

}

// Return the number of unique elements

return k;

}

// Main function to test the remove Duplicates function

int main() {

//code

printf("Array after removing duplicates: ");

for (i = 0; i < newSize; i++) {

printf("%d ", nums[i]);

}

printf("\n");

printf("Number of unique elements: %d\n", newSize);

return 0;

}

1. **Find the Index of first occurrence of a string**

#include <stdio.h>

#include <string.h>

int main() {

char haystack[100];

char needle[100];

scanf("%s", haystack);

scanf("%s", needle);

int result = strStr(haystack, needle);

printf("%d\n", result);

return 0;

}

int strStr(char\* haystack, char\* needle) {

int hsize = strlen(haystack);

int nsize = strlen(needle);

int res =-1;

int i = 0, j= 0;

//code

}