**PART A (MAIN PROGRAM)**

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

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

return 0;

}

1. **Playing with characters**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

return 0;

}

1. **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); }

return 0;

}

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; }

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

data = realloc(data, new\_length);

if (!data) { break; }

alloc\_length = new\_length;

}

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

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

}

data = realloc(data, data\_length);

return data;

}

1. **Printing patterns using loops**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

int n;

scanf("%d", &n);

return 0;

}

1. **Correctness and the Loop Invariant**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include <assert.h>

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

}

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 <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\*);

int parse\_int(char\*);

int solve(int n) {

}

int main()

{

    FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

    int t = parse\_int(ltrim(rtrim(readline())));

    for (int t\_itr = 0; t\_itr < t; t\_itr++) {

        int n = parse\_int(ltrim(rtrim(readline())));

        int result = solve(n);

        fprintf(fptr, "%d\n", result);

}

    fclose(fptr);

    return 0;

}

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;

}

int parse\_int(char\* str) {

    char\* endptr;

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

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

        exit(EXIT\_FAILURE);

    }

    return value;

}

1. **For loop in C**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

    int a, b;

    scanf("%d\n%d", &a, &b);

    return 0;

}

1. **Calculate the nth term**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

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

{

// write your code here

return term;

}

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. **Student Marks Sum**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int marks\_summation(int\* marks, int number\_of\_students, char gender)

{

// write your code here

}

int main()

{

int number\_of\_students;

char gender;

int sum;

scanf("%d", &number\_of\_students);

int \*marks = (int \*) malloc(number\_of\_students \* sizeof (int));

for (int student = 0; student < number\_of\_students; student++) {

scanf("%d", (marks + student));

}

scanf(" %c", &gender);

sum = marks\_summation(marks, number\_of\_students, gender);

printf("%d", sum);

free(marks);

return 0;

}

1. **SORTING ARRAY OF STRINGS**

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

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;

}

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

{

int i;

// write your code here

}

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, sort\_by\_length);

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

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

printf("\n");

}

1. **1D ARRAYS IN C**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n;

scanf("%d", &n);

// write your code here

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

free(arr);

return 0;

}

1. **Array Reversal**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int num, \*arr, i;

scanf("%d", &num);

// write your code here

return 0;

}

1. **Permutations of string**

#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;

}

//write your code here

}

int main()

{

char \*\*s;

int n,i;

scanf("%d", &n);

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

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

{

}

do

{

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

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

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;

}

1. **2D Arrays**

#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\*);

int main()

{

// write your code here

return 0;

}

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);

1if (endptr == str || \*endptr != '\0') {

exit(EXIT\_FAILURE);

}

return value;

}

**15. Dynamic Array in C**

#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;

//write your code here

}

}

// 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;

}