**Module 5**

**1. 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;

}

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;

}

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));

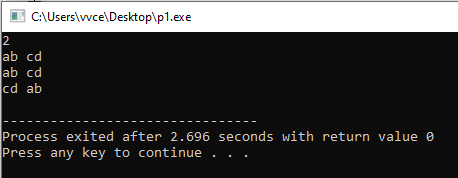
for ( i = 0; i < n; i++)

free(s[i]);

free(s);

return 0;

}

****

**2. 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\*);

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 ;

}

printf("%d",temp);

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;

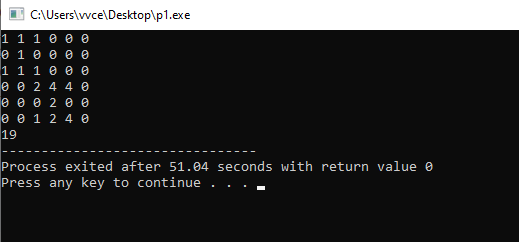
int value = strtol(str, &endptr, 10);

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

exit(EXIT\_FAILURE);

} return value;

}

****

**3. 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;

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]);

}

}

// Free allocated memory

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

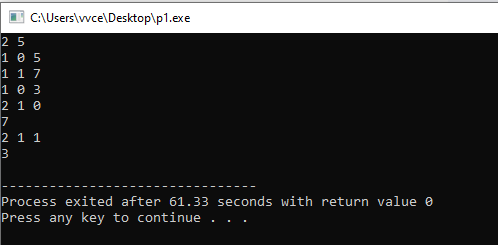
free(shelves[i]); // Free each shelf

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

free(sizes); // Free the sizes array

return 0;

}

****

**4. 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);

for(i=0;i<strlen(s);i++){

if(\*(s+i)==’ ‘)

printf(“\n”);

else

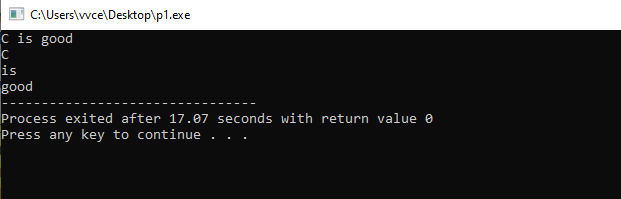
printf(“%c”,\*(s+i));

}

free(s);

return 0;

}

****

**5. Index of first occurrence of a string (Leetcode)**

#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;

while (haystack[i]!='\0' && needle[j]!='\0' ) {

if (haystack[i] == needle[j]) {

i++; j++;

}

else {

i++; j = 0;

}} if (

j

== nsize)

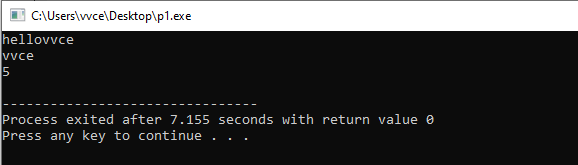
res =(i- nsize);

else

res=-1;

return res;

}

****