**1.Write a C program to reverse a given string without using any additional library functions.**

#include <stdio.h>

int stringLength(char str[]) {

int length = 0;

while (str[length] != '\0') {

length++;

}

return length;

}

void reverseString(char str[]) {

int length = stringLength(str);

int i, j;

char temp;

for (i = 0, j = length - 1; i < j; i++, j--) {

temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

int main() {

char str[100];

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

for (int i = 0; i < sizeof(str); i++) {

if (str[i] == '\n') {

str[i] = '\0';

break;

}

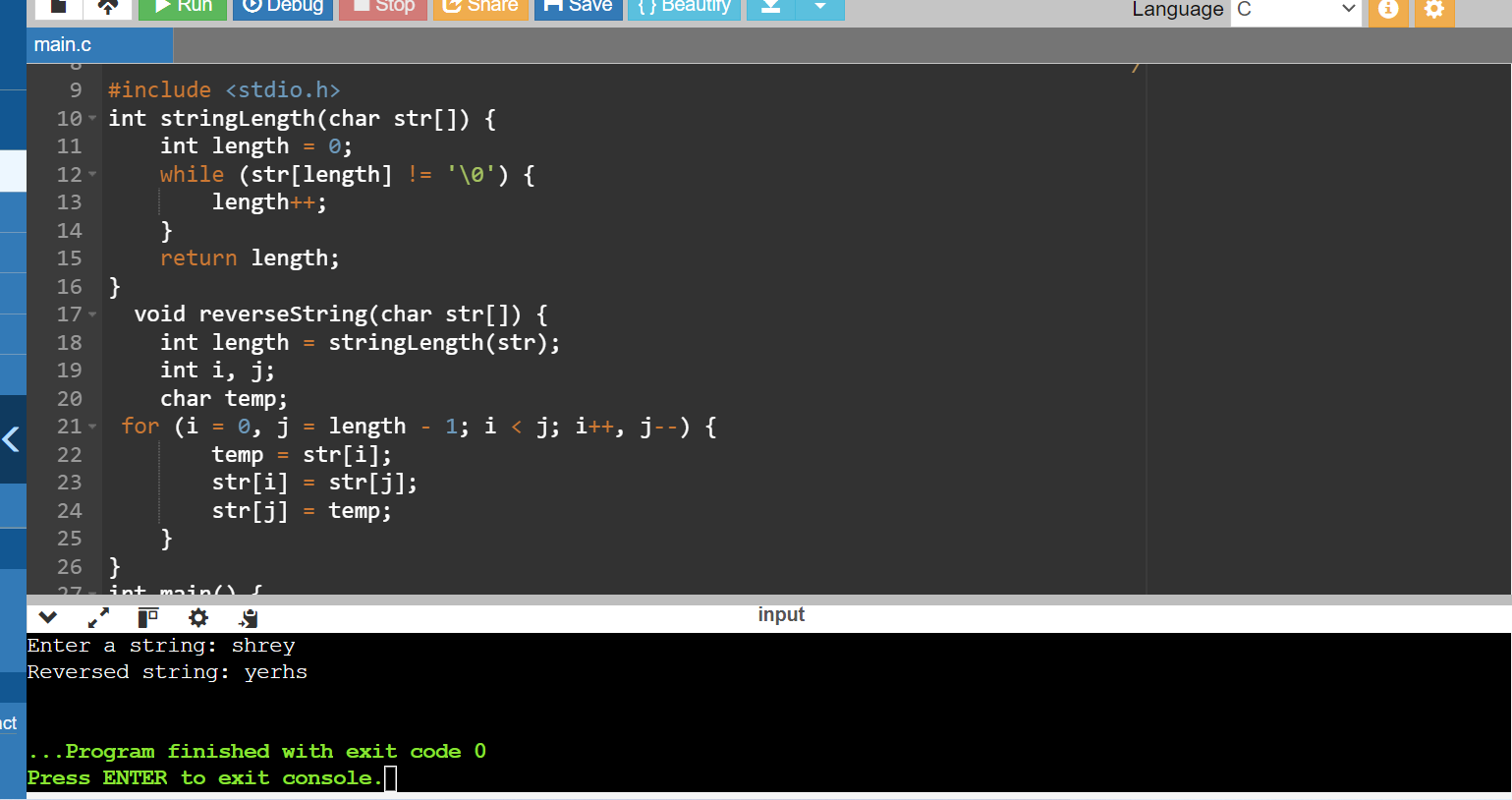
}

reverseString(str);

printf("Reversed string: %s\n", str);

return 0;

}



**2.Implement a function in C that takes an integer array and its size as arguments and sorts the elements of the array in ascending order using a selection sort algorithm.**

#include <stdio.h>

void selectionSort(int arr[], int n) {

int i, j, min\_idx, temp;

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

min\_idx = i;

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

if (arr[j] < arr[min\_idx]) {

min\_idx = j;

}

}

temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

void printArray(int arr[], int size) {

int i;

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

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

}

printf("\n");

}

int main() {

int arr[] = {56,33,21,11,78,89};

int n = sizeof(arr)/sizeof(arr[0]);

printf("Original array: \n");

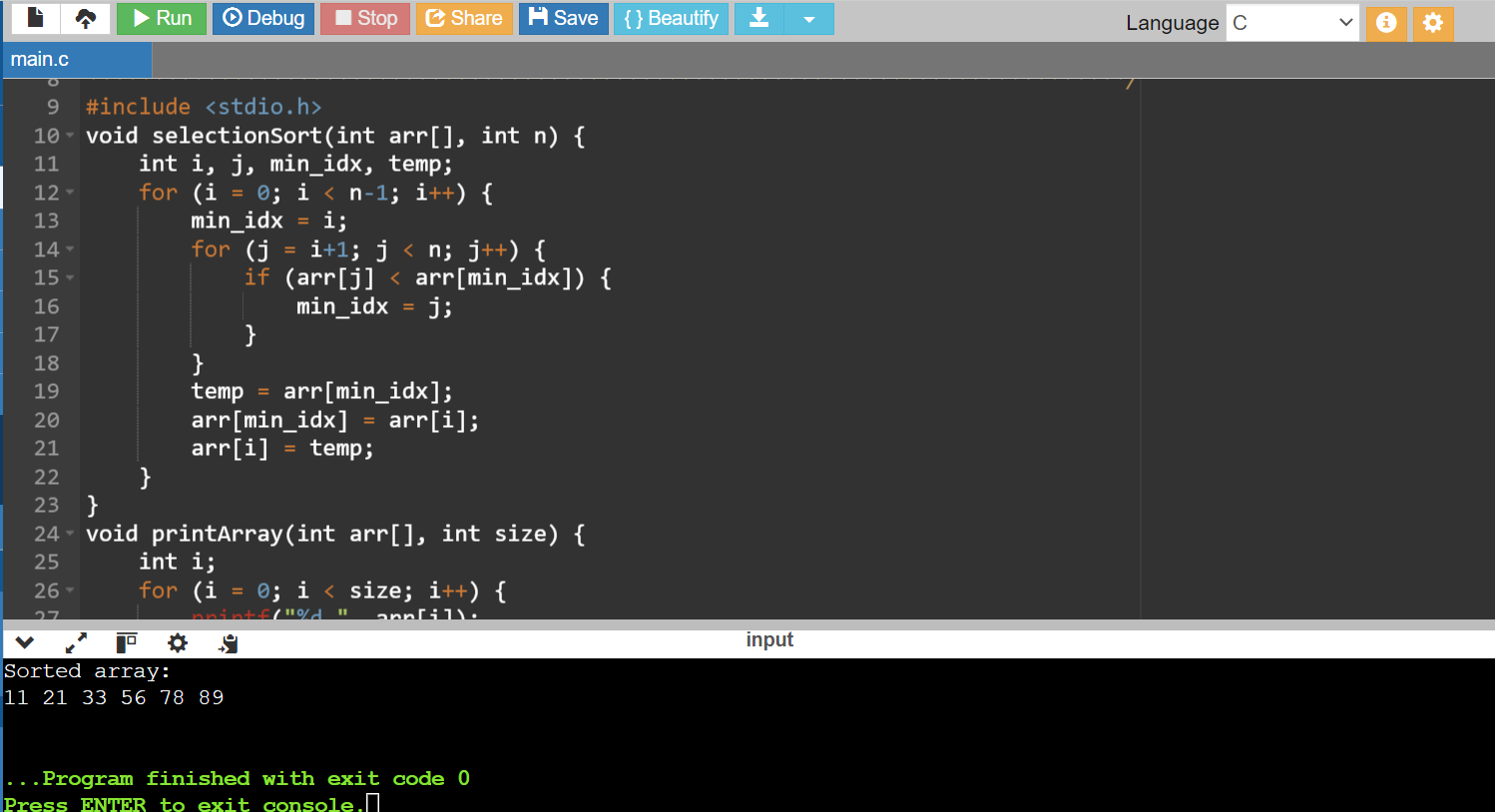
printArray(arr, n);

selectionSort(arr, n);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}

**Q.3 Explain the concept of structures in C and write a program to store student information (name, roll number, marks) using a structure**

The structure in C is a user-defined data type that can be used to group items of possibly different types into a single type. The **struct keyword**is used to define the structure in the C programming language. The items in the structure are called its **member** and they can be of any valid data type.To use structure in our program, we have to define its instance. We can do that by creating variables of the structure type**.**

We can define structure variables using two methods:

1.Structure Variable Declaration with Structure Template

2.Structure Variable Declaration after Structure Template

We can initialize structure members in 3 ways which are as follows:

1.Using Assignment Operator.

2.Using Initializer List.

3.Using Designated Initializer List.

#include <stdio.h>

struct Student {

char name[50];

int rollNumber;

float marks;

};

int main() {

struct Student student;

printf("Enter student name: ");

fgets(student.name, sizeof(student.name), stdin);

printf("Enter roll number: ");

scanf("%d", &student.rollNumber);

printf("Enter marks: ");

scanf("%f", &student.marks);

printf("\n Student Information:\n");

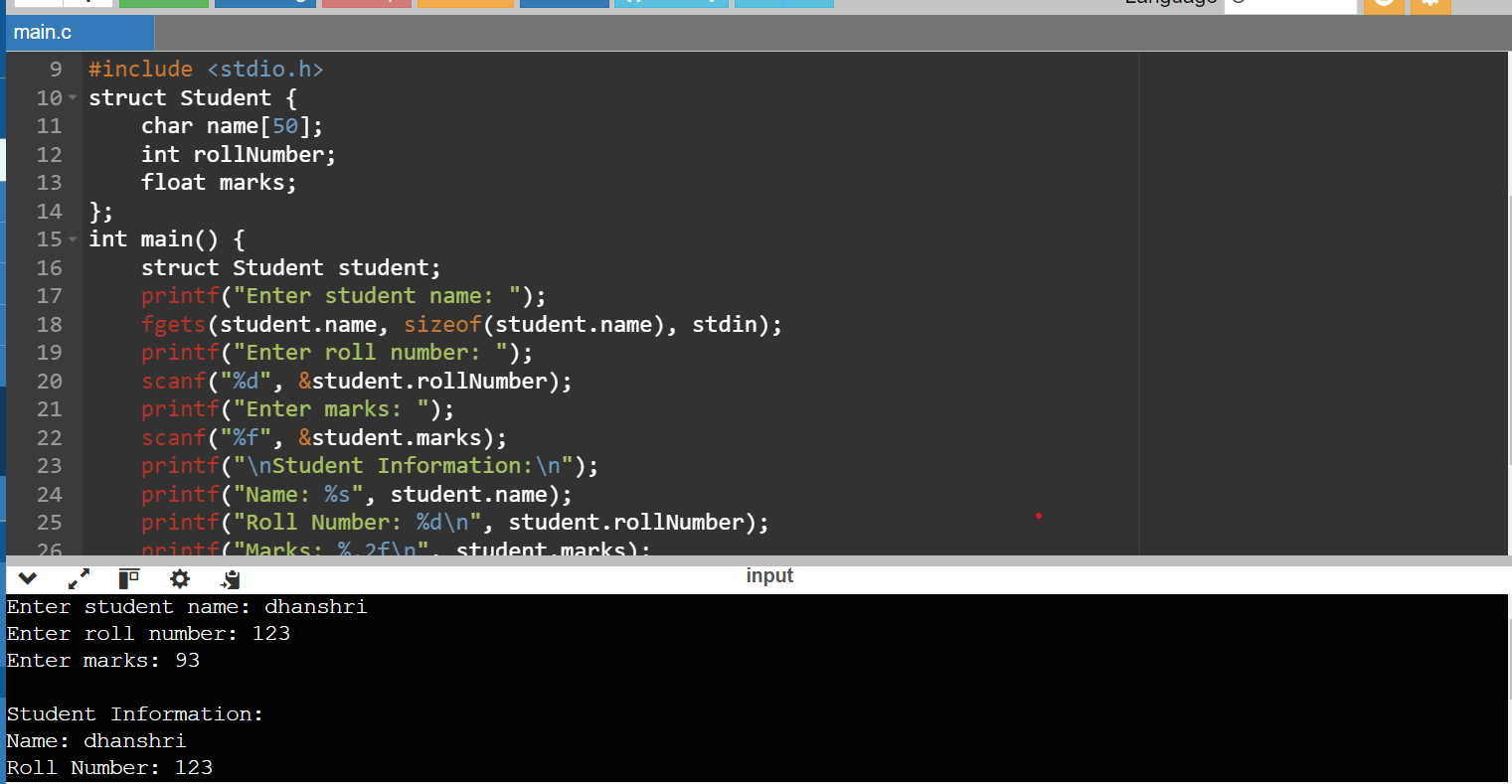
printf("Name: %s", student.name);

printf("Roll Number: %d\n", student.rollNumber);

printf("Marks: %.2f\n", student.marks);

return 0;

}



**Q.4 Differentiate between single-linked lists and doubly-linked lists in C. Write code snippets to create a node and perform a basic insertion operation in a singly-linked list.**

**Single-Linked Lists**:

* Each node contains data and a pointer to the next node.
* Traversal is only possible in one direction, i.e., from the head to the end of the list.
* Requires less memory per node compared to doubly-linked lists because it only has one pointer.
* Simpler to implement than doubly-linked lists.

**Doubly-Linked Lists**:

* Each node contains data, a pointer to the next node, and a pointer to the previous node.
* Traversal is possible in both directions, i.e., from the head to the end and from the end back to the head.
* Requires more memory per node because it has two pointers (next and prev).
* More complex to implement due to the additional pointer and need to update both pointers during insertion and deletion operations.

**Q.5 Explain the concept of pointers in C and write a program to swap the values of two variables using pointers.**

In C, a pointer is a variable that stores the memory address of another variable. allows direct manipulation of memory locations, enabling efficient access and manipulation of data. Pointers are extensively used for dynamic memory allocation, passing addresses to functions, and implementing data structures like linked lists and trees.

1. **Declaration**: A pointer is declared using the \* (asterisk) symbol before the variable name, indicating that it holds an address.

int \*ptr;

1. **Address-of Operator (&)**: This operator returns the memory address of a variable.

int x = 10;

int \*ptr = &x;

1. **Dereference Operator (\*)**: This operator is used to access the value stored at the address held by a pointer.

int y = \*ptr;

1. **Pointer Arithmetic**: Pointers can be incremented or decremented to navigate through memory locations.

ptr++;

1. **Null Pointers**: Pointers can also hold a special value NULL, which indicates that they are not pointing to any valid memory location.

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int x = 5, y = 10;

printf("Before swapping:\n");

printf("x = %d, y = %d\n", x, y);

swap(&x, &y);

printf("After swapping:\n");

printf("x = %d, y = %d\n", x, y);

return 0;

}

