// A function to sort the algorithm using gnome sort

void gnomeSort(int arr[], int n)

{

int index = 0;

while (index < n) {

if (index == 0)

index++;

if (arr[index] >= arr[index - 1])

index++;

else {

swap(arr[index], arr[index - 1]);

index--;

}

}

return;

}

// Function to find the word with the highest number of repeated letters

string highest\_repeated\_letters(string str) {

int str\_size = str.length(); // Store the length of the input string

int ctr1 = 0, ctr2 = 0, high1 = 0, high2 = 0; // Initializing counters and variables

int start = -1, end = 0; // Initializing start and end indices

int temp1, temp2; // Temporary variables for storing indices

char letter; // Variable to store letters

for (int x = 0; x < str\_size; x++, end++) // Loop through the input string

{

if (start == -1) // If start is not initialized

{

start = x; // Set the start index

}

if (str[x] == ' ' || x == str\_size - 1) // If a word is encountered (space or end of string)

{

if (end == str\_size - 1) // Handle case when the last word doesn't end with space

{

end += 1; // Adjust the end index

}

for (int y = start; y < end; y++) // Loop within the word boundaries

{

letter = str[y]; // Get the letter at index y

for (int z = start; z < end; z++) // Loop within the word to check for repeated letters

{

if (y == z)

{

continue; // Skip comparison with the same letter

}

else if (letter == str[z]) // If letters match, count repetition

{

ctr1++;

}

}

if (ctr1) // If there are repetitions

{

ctr2++; // Increase the counter for repeated letters

}

// Check if the current word has the highest number of repeated letters

if (ctr1 > high1 && ctr2 > high2)

{

high1 = ctr1;

high2 = ctr2;

temp1 = start;

temp2 = end;

}

ctr1 = 0; // Reset the letter counter for the next iteration

}

// Update variables for the word with the highest number of repeated letters

if (ctr2 > high2)

{

high2 = ctr2;

temp1 = start;

temp2 = end;

}

ctr2 = 0; // Reset the repeated letters counter for the next word

start = end + 1; // Update the start index for the next word

}

}

if (high1 > 0) // If a word with repeated letters is found

{

string repeated;

// Extract the word with the highest number of repeated letters

for (temp1; temp1 < temp2; temp1++)

{

repeated.push\_back(str[temp1]);

}

return repeated; // Return the word with the highest number of repeated letters

}

else

{

return "-1"; // Return -1 if no word with repeated letters is found

}

}

void towerOfHanoi(int n, char from\_rod, char to\_rod,

char aux\_rod)

{

if (n == 0) {

return;

}

towerOfHanoi(n - 1, from\_rod, aux\_rod, to\_rod);

cout << "Move disk " << n << " from rod " << from\_rod

<< " to rod " << to\_rod << endl;

towerOfHanoi(n - 1, aux\_rod, to\_rod, from\_rod);

}