**COMPUTER SCIENCE DEPARTMENT**

|  |
| --- |
| **Total Marks:** |
| **Obtained Marks:** |

Operating System Lab

**LAB TASK#12**

**Last date of Submission: May 2, 2025**

**Submitted To: Sir Jawad Naseer**

**Student Name: Ubaid Bin Waris**

**Reg Number: 2212416**

**COMPUTER SCIENCE DEPARTMENT**

**Question no.1**

**You're developing a budget tracker where users initially enter 7 daily expenses. The program uses calloc() to initialize and store the expenses. After entering all expenses, the user decides to update their data and reduce the number of tracked days to 4. Write a C program that:**

**a. Allocates memory for 7 float values using calloc().**

**b. Accepts user input for expenses.**

**c. Shrinks the memory size using realloc() to hold only the first 4 expenses.**

**d. Prints the updated expenses list.**

**e. Frees the allocated memory.**

**Code**

#include <stdio.h>

#include <stdlib.h>

int main() {

int original\_days = 7;

int updated\_days = 4;

float \*expenses;

// a. Allocate memory for 7 float values using calloc()

expenses = (float \*)calloc(original\_days, sizeof(float));

if (expenses == NULL) {

printf("Memory allocation failed.\n");

return 1;

}

// b. Accept user input for expenses

printf("Enter expenses for %d days:\n", original\_days);

for (int i = 0; i < original\_days; i++) {

printf("Day %d: ", i + 1);

scanf("%f", &expenses[i]);

}

// c. Shrink the memory size using realloc() to hold only the first 4 expenses

float \*temp = (float \*)realloc(expenses, updated\_days \* sizeof(float));

if (temp == NULL) {

printf("Memory reallocation failed.\n");

free(expenses); // still free the original block if realloc fails

return 1;

}

expenses = temp;

// d. Print the updated expenses list

printf("\nUpdated expenses for %d days:\n", updated\_days);

for (int i = 0; i < updated\_days; i++) {

printf("Day %d: %.2f\n", i + 1, expenses[i]);

}

// e. Free the allocated memory

free(expenses);

return 0;

}

**Output**

A screen shot of a computer program

AI-generated content may be incorrect.

**Question no.2**

**Imagine a file management system that starts with space for 100 file names (strings), but halfway through, the user decides they only need space for 30 files. Write a C program that:**

**a. Allocates memory for 100 file name entries using calloc().**

**b. Simulates name input for the first 30 entries.**

**c. Shrinks the memory to only hold 30 entries using realloc().**

**d. Displays the list and ensures memory is freed properly.**

**Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_FILES 100

#define USED\_FILES 30

#define MAX\_NAME\_LEN 100

int main() {

char \*\*fileNames;

// a. Allocate memory for 100 file name entries using calloc()

fileNames = (char \*\*)calloc(MAX\_FILES, sizeof(char \*));

if (fileNames == NULL) {

printf("Memory allocation for file pointers failed.\n");

return 1;

}

// b. Simulate name input for the first 30 entries

for (int i = 0; i < USED\_FILES; i++) {

fileNames[i] = (char \*)calloc(MAX\_NAME\_LEN, sizeof(char));

if (fileNames[i] == NULL) {

printf("Memory allocation for file name %d failed.\n", i + 1);

// Clean up already allocated memory

for (int j = 0; j < i; j++) {

free(fileNames[j]);

}

free(fileNames);

return 1;

}

// Simulate input (you can replace with scanf if real input is desired)

snprintf(fileNames[i], MAX\_NAME\_LEN, "file\_%d.txt", i + 1);

}

// c. Shrink the memory to only hold 30 entries using realloc()

char \*\*temp = (char \*\*)realloc(fileNames, USED\_FILES \* sizeof(char \*));

if (temp == NULL) {

printf("Memory reallocation failed.\n");

// Still free individual file names

for (int i = 0; i < USED\_FILES; i++) {

free(fileNames[i]);

}

free(fileNames);

return 1;

}

fileNames = temp;

// d. Display the list and ensure memory is freed properly

printf("File list after resizing to %d entries:\n", USED\_FILES);

for (int i = 0; i < USED\_FILES; i++) {

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

free(fileNames[i]); // Free each file name string

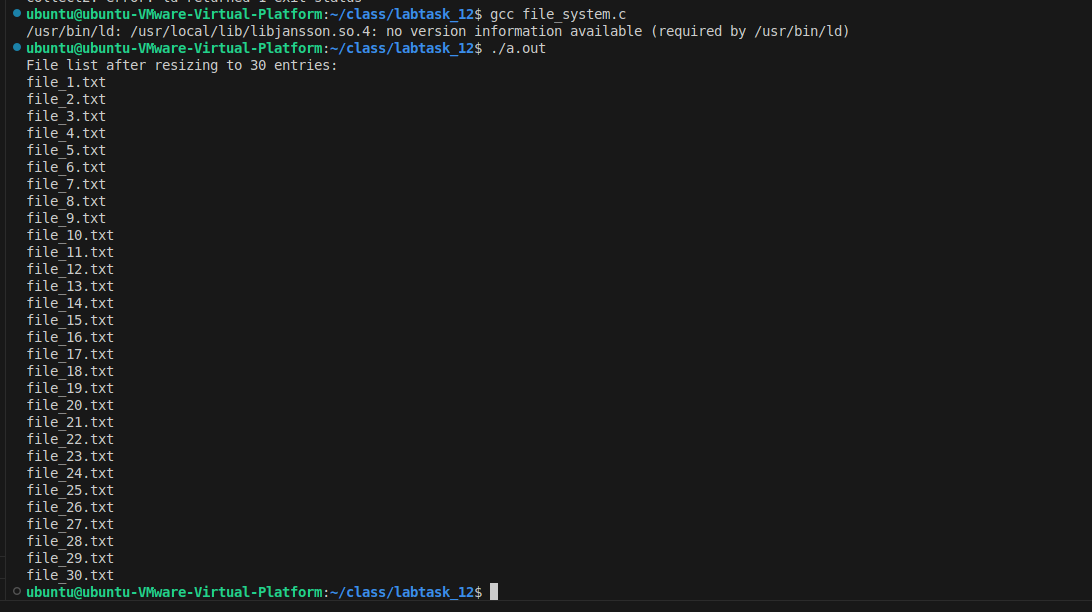
}

free(fileNames); // Free the array of pointers

return 0;

}

**Output**



**Question no.3**

**A developer wrote a program to allocate memory using malloc for a list of 6 product prices. However, they forgot to free the memory after use, causing a memory leak. Write a C program that:**

**a. Dynamically allocates memory for 6 float product prices.**

**b. Accepts user input and calculates the average price.**

**c. Corrects the bug by freeing the memory at the end and explains what could go wrong if it’s not freed.**

**Code**

#include <stdio.h>

#include <stdlib.h>

int main() {

int count = 6;

float \*prices;

float sum = 0.0, average;

// a. Dynamically allocate memory for 6 float product prices

prices = (float \*)malloc(count \* sizeof(float));

if (prices == NULL) {

printf("Memory allocation failed.\n");

return 1;

}

// b. Accept user input and calculate the average price

printf("Enter prices of %d products:\n", count);

for (int i = 0; i < count; i++) {

printf("Price %d: ", i + 1);

scanf("%f", &prices[i]);

sum += prices[i];

}

average = sum / count;

printf("\nAverage Price: %.2f\n", average);

// c. Free the allocated memory

free(prices);

return 0;

}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

**Question no.4**

**A sensor records temperature values throughout the day. However, the memory is allocated without knowing how many readings will come in.**

**Your task is to write a C program that:**

**a. Continuously accepts temperature readings one by one.**

**b. For each new reading, use realloc to increase the storage.**

**c. Stop when the user enters -1.**

**d. Display all stored readings and release the memory properly using free.**

**Code**

#include <stdio.h>

#include <stdlib.h>

int main() {

float \*temperatures = NULL;

int count = 0;

float input;

printf("Enter temperature readings one by one (enter -1 to stop):\n");

while (1) {

printf("Reading %d: ", count + 1);

scanf("%f", &input);

if (input == -1.0f)

break;

// b. Reallocate memory to store one more reading

float \*temp = (float \*)realloc(temperatures, (count + 1) \* sizeof(float));

if (temp == NULL) {

printf("Memory reallocation failed.\n");

free(temperatures);

return 1;

}

temperatures = temp;

temperatures[count] = input;

count++;

}

// d. Display all stored readings

printf("\nStored Temperature Readings:\n");

for (int i = 0; i < count; i++) {

printf("Reading %d: %.2f\n", i + 1, temperatures[i]);

}

// Free allocated memory

free(temperatures);

return 0;

}

**Output**

A computer screen with white text

AI-generated content may be incorrect.