Assignment - 22 (DMA)

1. Define a function to input variable length string and store it in an array without

memory wastage.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void inputString()

{

// char \*s = (char \*)calloc(strlen(s), sizeof(char)); // string for input

char \*s, \*ptr;

int i;

printf("Enter a string: ");

gets(s);

for(i = 0; s[i]; i++);

// fgets(s, strlen(s), stdin);

// s[strlen(s) - 1] = '\0';

ptr = (char \*)calloc(i, sizeof(char)); // string in which entered string is to be copied.

strcpy(ptr, s);

printf("String entered as input string is %s.\n", ptr);

}

int main()

{

inputString();

return 0;

}

2. Write a program to ask the user to input a number of data values he would like to enter then create an array dynamically to accommodate the data values. Now take the input from the user and display the average of data values.

#include<stdio.h>

#include<stdlib.h>

int main()

{

int size, \*arr, i;

double sum = 0;

printf("Enter number of data values you want to enter: ");

scanf("%d", &size);

arr = calloc(size, sizeof(int));

printf("Enter %d values that are to be stored in the array:\n", size);

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

{

scanf("%d", &arr[i]);

}

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

{

sum += arr[i];

}

printf("Average of entered data values is %lf.", sum / size);

return 0;

}

3. Write a program to calculate the sum of n numbers entered by the user using malloc and free.

#include<stdio.h>

#include<stdlib.h>

int main()

{

int n, i;

double sum = 0, \*var = NULL;

printf("Enter number of values that you want to enter: ");

scanf("%d", &n);

printf("Enter %d values:\n", n);

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

{

var = (double \*)malloc(sizeof(double));

scanf("%lf", var);

sum += \*var;

free(var);

}

printf("Sum of %d numbers is %lf.", n, sum);

return 0;

}

4. Write a program to input and print text using dynamic memory allocation.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

char \*text = NULL;

text = (char \*)calloc(100, sizeof(char));

printf("Enter a string: ");

fgets(text, 100, stdin);

text[strlen(text) - 1] = '\0';

printf("\nEntered text is %s\n", text);

return 0;

}

5. Write a program to read a one dimensional array, print sum of all elements along with inputted array elements using dynamic memory allocation.

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*arr = NULL, size, i, sum = 0;

printf("Enter size of array: ");

scanf("%d", &size);

arr = (int \*)calloc(size, sizeof(int));

printf("Enter %d elements of array:\n", size);

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

{

scanf("%d", &arr[i]);

}

printf("Entered array elements are:\n");

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

{

sum += arr[i];

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

}

printf("\nSum of array elements is %d.\n", sum);

return 0;

}

6. Write a program in C to find the largest element using Dynamic Memory Allocation.

#include<stdio.h>

#include<stdlib.h>

int main()

{

//int arr[10], i, largest;

int \*arr, size, i, largest;

printf("Enter number of elements that you want to store in array: ");

scanf("%d", &size);

arr = (int \*)calloc(size, sizeof(int));

printf("Enter %d elements of array:\n", size);

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

scanf("%d", &arr[i]);

largest = arr[0];

for(i = 1; i < size; i++)

if(arr[i] > largest)

largest = arr[i];

printf("Greatest element in array is %d.", largest);

return 0;

}

7. Write a program to demonstrate memory leak in C.

#include<stdio.h>

#include<stdlib.h>

void printValueofDMAVar()

{

int \*ptr = NULL;

ptr = (int \*)malloc(sizeof(int));

\*ptr = 5;

printf("Value of DMA variable is %d.", \*ptr);

/\* after this line function ends but DMA variable memory does not gets released and local variable ptr which was

pointing to this DMA variable gets destroyed when this function ends. Now because pointer variable pointing to this

DMA variable gets destroyed, we lost the address of DMA variable. Hence, memory leak occurs. \*/

}

int main()

{

printValueofDMAVar();

return 0;

}

8. Write a program to demonstrate dangling pointers in C.

#include<stdio.h>

#include<stdlib.h>

void printValueOfDMAVariable()

{

int \*p = NULL;

p = (int \*)malloc(sizeof(int));

\*p = 5;

printf("Value of DMA variable is %d.", \*p);

free(p); // Memory of DMA variable gets released

/\*

Now p has become dangling pointer because the memory location which p was initially pointing to got destroyed

by free() function.

\*/

/\*

the original DMA variable got destroyed because of free(), so now p is pointing to location which has no

significance. Hence, now here it is illegal memory access

\*/

printf("\nValue of DMA variable is %d.\n", \*p);

}

int main()

{

printValueOfDMAVariable();

return 0;

}

9. Write a program to allocate memory dynamically of the size in bytes entered by the user. Also handle the case when memory allocation is failed.

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*arr, size, i;

printf("Enter number of elements that are to be stored in array: ");

scanf("%d", &size);

arr = (int \*)calloc(size, sizeof(int));

if(arr == NULL)

{

printf("Memory Allocation Failed!!!\n");

return 0;

}

printf("Array of size %d created dynamically.\n\n", size);

printf("Enter %d values that are to be stored in the array:\n", size);

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

{

scanf("%d", &arr[i]);

}

printf("\n%d array elements are:\n", size);

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

{

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

}

free(arr);

return 0;

}

10. Find out the maximum and minimum from an array using dynamic memory allocation in C.

#include<stdio.h>

#include<stdlib.h>

int main()

{

int size, \*arr = NULL, i, largest, smallest;

printf("Enter number of data values you want to enter: ");

scanf("%d", &size);

arr = (int \*)calloc(size, sizeof(int));

if(arr == NULL)

{

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

return 0;

}

printf("Enter %d values that are to be stored in the array:\n", size);

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

{

scanf("%d", &arr[i]);

}

largest = arr[0];

for(i = 1; i < size; i++)

if(arr[i] > largest)

largest = arr[i];

smallest = arr[0];

for(i = 1; i < size; i++)

if(arr[i] < smallest)

smallest = arr[i];

printf("\nSmallest element in array is %d.\n", smallest);

printf("Greatest element in array is %d.", largest);

free(arr);

return 0;

}