SEMESTER:-02

EXERCISE:- 01

Q1] Input:-

#include <stdio.h>

int main(void) {

int \*ptr;

int x;

ptr = &x;

\*ptr = 0;

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

printf("\*ptr = %d\n", \*ptr);

\*ptr +=5;

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

printf("\*ptr = %d\n", \*ptr);

(\*ptr)++;

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

printf("\*ptr = %d\n", \*ptr);

return 0;

}

Output:-

x = 0

\*ptr = 0

x = 5

\*ptr = 5

x = 6

\*ptr = 6

Q2] using pointers and without using pointers recursion:-

|  |  |  |
| --- | --- | --- |
| // without using pointers Input:-  #include <stdio.h>  void square(int n);  int main(void) {  int number=4;  square(number);  printf("number=%d\n", number);  return 0; }  void square(int n) {  n=(n) \* (n);  printf("square=%d\n", n);  }  Output:- square=16  number=4 | // with using pointers Input:-  #include <stdio.h>  void square(int \*n);  int main(void) {  int number=4;  square(&number);  printf("number=%d\n", number);  return 0; }  void square(int \*n) {  \*n=(\*n) \* (\*n);  printf("square=%d\n", \*n);  }  Output:- square=16  number=16 |  |

## SET A :-

Q1:- Input:-

#include <stdio.h>

int main() {

int n; //do not declare array here

// Input the number of elements

printf("Enter the number of elements: ");

scanf("%d", &n);

// Check if n is a positive integer

if (n <= 0) {

printf("Please enter a positive integer for the number of elements.\n");

return 1; // Exit the program with an error code

}

int arr[n]; // declare array saperately

// Input elements from the user

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

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

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

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

}

// Display elements in reverse order

printf("\nElements in reverse order:\n");

for (int i = n - 1; i >= 0; i--) {

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

}

return 0; // Exit the program successfully

}

Output:-

Enter the number of elements: 3

Enter 3 elements:

Element 1: 1

Element 2: 2

Element 3: 3

Elements in reverse order:

3

2

1

Enter the number of elements: -1

Please enter a positive integer for the number of elements.

### OR

#include <stdio.h>

void enternum(int arr[],int n);

void printnum(int arr[],int n);

int main() {

int n;

printf("enter the number of elements:");

scanf("%d",&n);

int arr[n];

enternum(arr,n);

printnum(arr,n);

return 0;

}

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

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

printf("enter the element %d:",i+1);

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

}

}

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

for(int i=n-1;i>=0;i--) {

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

}

Output:-

enter the number of elements:3

enter the element 1:1

enter the element 2:2

enter the element 3:3

3 2 1

HINT:-

The error "non-void function does not return a value" typically occurs when a function is declared to return a value, but not all code paths within the function contain a return statement. To fix this, you need to ensure that all code paths within the function return a value.

For example, in file func.c at line 7, if the function is declared to return a non-void type such as int, then all possible code paths within the function need to include a return statement that returns a value of type int.

Or

#include<stdio.h>

void reverse\_order(int array[],int n);

int main() {

int n;

printf("enter the size of the array:- ");

scanf("%d",&n);

int array[n];

printf("enter the %d elements in the array:- \n",n);

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

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

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

}

reverse\_order(array,n);

}

void reverse\_order(int array[],int n) {

int a=1;

printf("the reverse order of the array is:- \n");

for(int i=n-1;i>=0;i--) {

printf("%d value :- %d\n",a++,array[i]);

}

}

Q2:- Input:-

#include <stdio.h>

// Function for Linear Search

int linearSearch(int arr[], int n, int key) {

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

if (arr[i] == key) {

return i; // Key found at position i

}

}

return -1; // Key not found

}

int main() {

int n;

// Get the number of elements

printf("Enter the number of elements: ");

scanf("%d", &n);

// Declare an array of size n

int numbers[n];

// Accept n numbers into the array

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

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

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

}

// Accept the key to be searched

int key;

printf("Enter the key to be searched: ");

scanf("%d", &key);

// Call linearSearch function to find the position of the key

int position = linearSearch(numbers, n, key);

// Display appropriate messages

if (position != -1) {

printf("Key %d found at position %d.\n", key, position + 1); // Adding 1 to convert to human-readable position

} else {

printf("Key %d not found in the array.\n", key);

}

return 0;

}

Output:-

Enter the number of elements: 3

Enter 3 numbers:

22

33

44

Enter the key to be searched: 3

Key 3 not found in the array.

Enter the number of elements: 3

Enter 3 numbers:

1

2

3

Enter the key to be searched: 2

Key 2 found at position 2.

### OR

#include<stdio.h>

int linearkeysearch(int arr[],int n,int key);

int main() {

int n;

printf("enter the number of elements:");

scanf("%d",&n);

int numbers[n];

printf("enter %d elements\n",n);

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

printf("the element %d:-",i+1);

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

}

int key;

printf("enter the key to be searched:");

scanf("%d",&key);

int position=linearkeysearch(numbers,n,key);

if(position!=-1) {

printf("key %d found at position %d",key,position+1);

}

else {

printf("key %d not found in the array",key);

}

return 0;

}

int linearkeysearch(int arr[],int n,int key) {

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

if(arr[i]==key) {

return i;

}

}

return -1;

}

Output:-

enter the number of elements:4

enter 4 elements

the element 1:-2

the element 2:-3

the element 3:-4

the element 4:-6

enter the key to be searched:4

key 4 found at position 3

enter the number of elements:3

enter 3 elements

the element 1:-6

the element 2:-8

the element 3:-9

enter the key to be searched:3

key 3 not found in the array

Q3:- Input:-

#include<stdio.h>

void bubble\_sort(int a[],int n);

int main() {

int n;

printf("enter the number of elements:-");

scanf("%d",&n);

int numbers[n];

printf("enter the elements\n");

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

printf("enter the %d element:-",i+1);

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

}

bubble\_sort(numbers,n);

printf("the sorted array is:-");

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

printf("%d\t",numbers[i]);

}

return 0;

}

void bubble\_sort(int a[],int n) {

int temp;

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

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

if(a[j]>a[j+1]) {

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

Output:-

enter the number of elements:-3

enter the elements

enter the 1 element:-9

enter the 2 element:-6

enter the 3 element:-7

the sorted array is:-6 7 9

Q4:- Input:-

#include<stdio.h>

void insertion\_sort(int arra[],int n);

int main() {

int num;

printf("how many numbers you wan't to sort:-");

scanf("%d",&num);

if(num<0) {

printf("enter a positive number");

}

int arr[num];

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

printf("enter the %d:-",i+1);

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

}

insertion\_sort(arr,num);

printf("the sorted array is :-");

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

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

}

}

void insertion\_sort(int arra[],int n) {

int temp,j;

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

temp=arra[i];

j=i-1;

while(j>=0 && arra[j]>temp) {

arra[j+1]=arra[j];

j--;

}

arra[j+1]=temp;

}

}

Output:-

how many numbers you wan't to sort:-10

enter the 1:-23

enter the 2:-567

enter the 3:-768

enter the 4:-3

enter the 5:-8

enter the 6:-3

enter the 7:-9

enter the 8:-398

enter the 9:-89

enter the 10:-5

the sorted array is :-3 3 5 8 9 23 89 398 567 768

Q5:- Input:-

#include <stdio.h>

int binary\_search(int arr[], int n, int key);

int main() {

int num,position,key;

printf("enter the size of the array: ");

scanf("%d",&num);

int arra[num];

printf("enter %d numbers\n",num);

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

printf("enter in assending order the %d number is: ",i+1);

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

}

printf("enter the key to be searched: ");

scanf("%d",&key);

position = binary\_search(arra,num,key);

if(position != -1) {

printf("the key %d is found at position %d",key,position+1);

}

else {

printf("the key %d is not found in the elements",key);

}

}

int binary\_search(int arr[],int n,int key) {

int left=0,right=n-1;

while(left<=right) {

int mid=(left+right)/2;

if(arr[mid]==key) {

return mid;

}

else if(key > arr[mid]) {

left=mid + 1;

}

else {

right = mid - 1;

}

}

return -1;

}

Output:-

enter the size of the array: 6

enter 6 numbers

enter in assending order the 1 number is: 1

enter in assending order the 2 number is: 2

enter in assending order the 3 number is: 3

enter in assending order the 4 number is: 4

enter in assending order the 5 number is: 5

enter in assending order the 6 number is: 6

enter the key to be searched: 6

the key 6 is found at position 6

Or

#include<stdio.h>

int binary\_search(int array[],int n,int key);

void sorted\_array(int array[],int n);

int main() {

int n,key;

printf("enter the size of the array:- ");

scanf("%d",&n);

int array[n];

printf("enter the %d elements:- \n",n);

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

printf("%d value = ",i+1);

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

}

sorted\_array(array,n);

printf("the sorted array is:- \n");

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

printf("%d value= %d\n",i+1,array[i]);

}

printf("Enter the key to be searched:- \n");

scanf("%d",&key);

int position=binary\_search(array,n,key);

if(position != -1) {

printf("The %d element is found as position:- %d",key,position);

}

else {

printf("The %d element not found in the array",key);

}

}

void sorted\_array(int array[],int n) {

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

int temp=array[i];

int j=i-1;

while(j>=0 && array[j]>temp) {

array[j+1]=array[j];

j--;

}

array[j+1]=temp;

}

}

int binary\_search(int array[],int n,int key) {

int l=0;

int r=n-1;

while(l<=r){

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

int mid=(l+r)/2;

if(array[mid] == key) {

return mid+1;

}

else if (array[mid] > key) {

return r=mid-1;

}

else {

return l=mid+1;

}

}

return -1;

}

return 0;

}

Output:-

enter the size of the array:- 8

enter the 8 elements:-

1 value = 1

2 value = 23

3 value = 56

4 value = 435

5 value = 2

6 value =

37

7 value = 3

8 value = 6

the sorted array is:-

1 value= 1

2 value= 2

3 value= 3

4 value= 6

5 value= 23

6 value= 37

7 value= 56

8 value= 435

Enter the key to be searched:-

6

The 6 element is found as position:- 4

EXERCISE :- 02

SET A:-

Q1] Input:-

#include<stdio.h>

void matrixA(int array[10][20],int m,int n);

void tranpose\_of\_matrix(int array[10][20],int m,int n);

int main() {

int m,n,i,j;

int array[10][20]; /\* ALL WAYS DEFINE THE SIZE FIRST WITH SOME INTEGER VALUES \*/

printf("Enter the number of rows and columns of matrix:- ");

scanf("%d %d",&m,&n);

array[m][n];

matrixA(array,m,n);

printf("The matrix is:-\n");

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

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

printf("%d\t",array[i][j]);

}

printf("\n");

}

tranpose\_of\_matrix(array,m,n);

return 0;

}

void tranpose\_of\_matrix(int array[10][20],int m,int n) {

int i,j;

printf("The transpose of the matrix (%dX%d) is (%dX%d):- \n",m,n,n,m);

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

for(j=0;j<m;j++) {

printf("%d\t",array[j][i]);

}

printf("\n");

}

}

void matrixA(int array[10][20],int m,int n){

int i,j;

printf("Enter the elements of matrix:- \n");

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

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

printf("Enter the element at (%d,%d):- ",i+1,j+1);

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

}

}

}

Output:-

Enter the number of rows and columns of matrix:- 2 3

Enter the elements of matrix:-

Enter the element at (1,1):- 1

Enter the element at (1,2):- 2

Enter the element at (1,3):- 3

Enter the element at (2,1):- 4

Enter the element at (2,2):- 5

Enter the element at (2,3):- 6

The matrix is:-

1 2 3

4 5 6

The transpose of the matrix (2X3) is (3X2):-

1 4

2 5

3 6

Q2]Input:-

#include<stdio.h>

void matriA(int arrayA[10][20],int m,int n);

void matriB(int arrayB[10][20],int x,int y);

void sumAB(int arrayA[10][20],int m,int n,int arrayB[10][20],int x,int y);

int main() {

int i,j,m,n,x,y,a,b;

int arrayA[10][20];

int arrayB[10][20];

printf("Enter the number of rows and columns of matrix A: ");

scanf("%d %d",&m,&n);

printf("Enter the number of rows and columns of matrix B: ");

scanf("%d %d",&x,&y);

if(m == x && n == y) {

printf("The nessary checks are done the matrix A & B can be added\n");

a=arrayA[m][n];

b=arrayB[x][y];

printf("Enter the elements of matrix A:- \n");

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

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

printf("value at (%d,%d):- ",i+1,j+1);

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

}

}

printf("Enter the elements of matrix B:- \n");

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

for(j=0;j<y;j++) {

printf("value at (%d,%d):- ",i+1,j+1);

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

}

}

matriA(arrayA,m,n);

matriB(arrayB,x,y);

sumAB(arrayA,m,n,arrayB,x,y);

}

else {

printf("(mXn) is not equal to (xXy)");

}

}

void matriA(int arrayA[10][20],int m,int n) {

int i,j;

printf("The matrix A is:-\n");

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

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

printf("%d\t",arrayA[i][j]);

}

printf("\n");

}

}

void matriB(int arrayB[10][20],int x,int y) {

int i,j;

printf("The matrix B is:-\n");

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

for(j=0;j<y;j++) {

printf("%d\t",arrayB[i][j]);

}

printf("\n");

}

}

void sumAB(int arrayA[10][20],int m,int n,int arrayB[10][20],int x,int y) {

int i,j;

int arrayC[10][20];

printf("The sum of matrix A & B is:-\n");

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

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

arrayC[i][j] = arrayA[i][j] + arrayB[i][j];

printf("%d\t",arrayC[i][j]);

}

printf("\n");

}

}

Output:-

Enter the number of rows and columns of matrix A: 2 2

Enter the number of rows and columns of matrix B: 2 2

The nessary checks are done the matrix A & B can be added

Enter the elements of matrix A:-

value at (1,1):- 1

value at (1,2):- 2

value at (2,1):- 3

value at (2,2):- 4

Enter the elements of matrix B:-

value at (1,1):- 5

value at (1,2):- 6

value at (2,1):- 7

value at (2,2):- 8

The matrix A is:-

1 2

3 4

The matrix B is:-

5 6

7 8

The sum of matrix A & B is:-

6 8

10 12

Exercise:-03

# Set A:-

Q1]input:-

#include<stdio.h>

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

int main() {

int A,B;

printf("Enter the value of A: ");

scanf("%d",&A);

printf("Enter the value of B: ");

scanf("%d",&B);

printf("Before swapping: A = %d, B = %d\n",A,B);

swap(&A,&B);

printf("After swapping: A = %d, B = %d\n",A,B);

}

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

int temp=\*a;

\*a=\*b;

\*b=temp;

}

Output:-

Enter the value of B: 3

Before swapping: A = 2, B = 3

After swapping: A = 3, B = 2

Q2] Input:-

#include<stdio.h>

int largest(int \*ptr,int n);

int main() {

int array[50];

int n;

printf("Enter the size of the array:- ");

scanf("%d",&n);

printf("Enter the %d elements :- \n",n);

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

printf("%d value is:- ",i+1);

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

}

int largestptr=largest(array,n);

printf("The largest number from the array using pointer is:- %d",largestptr);

}

int largest(int \*ptr,int n) {

int large=0;

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

if(\*(ptr+i)>large) {

large=\*(ptr+i);

}

}

return large;

}

Output:-

Enter the size of the array:- 4

Enter the 4 elements :-

1 value is:- 34

2 value is:- 457

3 value is:- 3

4 value is:- 78

The largest number from the array using pointer is:- 457

Q3] input:-

#include<stdio.h>

void reverseptr(int \*ptr,int n);

int main() {

int n;

int array[50];

printf("Enter the size of the array:- ");

scanf("%d",&n);

printf("Enter the %d elements:- ",n);

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

printf("%d value is:- ",i+1);

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

}

reverseptr(array,n);

}

void reverseptr(int \*ptr,int n) {

int j=1;

printf("The nubers in reverse order is:- \n");

for(int i=n-1;i>=0;i--) {

printf("%d value is:- %d\n",j++,\*(ptr+i));

}

}

Output:-

Enter the size of the array:- 8

Enter the 8 elements:- 1 value is:- 1

2 value is:- 2

3 value is:- 3

4 value is:- 4

5 value is:- 5

6 value is:- 6

7 value is:- 7

8 value is:- 8

The nubers in reverse order is:-

1 value is:- 8

2 value is:- 7

3 value is:- 6

4 value is:- 5

5 value is:- 4

6 value is:- 3

7 value is:- 2

8 value is:- 1

Exercise :-04

Q1]Input:- //Dynamic Memory allocation

#include<stdio.h>

#include<stdlib.h>

int main() {

int n;

printf("Enter the size to allocate the memory:- ");

scanf("%d",&n);

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

if(ptr==NULL) {

printf("Memory allocation failed");

}

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

ptr[i]=0;

}

printf("Allocated memory value set to zero:- \n");

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

printf("%d\n",ptr[i]);

}

printf("changing the value from zero:- \n");

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

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

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

}

printf("printing the changed value:- \n");

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

printf("%d\n",ptr[i]);

}

}

Output:-

Enter the size to allocate the memory:- 5

Allocated memory value set to zero:-

0

0

0

0

0

changing the value from zero:-

1 value:- 1

2 value:- 2

3 value:- 3

4 value:- 4

5 value:- 5

printing the changed value:-

1

2

3

4

5

Q2] Input:-

#include<stdio.h>

#include<stdlib.h>

int main() {

int n,min,max;

printf("Enter the size to allocate the memory:- ");

scanf("%d",&n);

int \*ptr=(int\*)calloc(n,sizeof(int));

if(ptr==NULL) {

printf("Memory allocation failed");

}

printf("Allocated memory value set to zero by calloc function:- \n");

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

printf("%d\n",ptr[i]);

}

printf("changing the value from zero to user input value:- \n");

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

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

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

}

printf("finding the range of the values\n");

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

min=ptr[0];

max=ptr[0];

if(ptr[i]>max) {

max=ptr[i];

}

if(ptr[i]<min) {

min=ptr[i];

}

}

printf("The range of the values are:- \n1st:- min=%d \n2nd:- max=%d",min,max);

}

Output:-

Enter the size to allocate the memory:- 4

Allocated memory value set to zero by calloc function:-

0

0

0

0

changing the value from zero to user input value:-

1 value:- 1

2 value:- 2

3 value:- 3

4 value:- 4

finding the range of the values

The range of the values are:-

1st:- min=1

2nd:- max=4

Q3] Input:-

#include <stdio.h>

#include <stdlib.h>

int main() {

int n;

printf("Enter the number of integers: ");

scanf("%d", &n);

// Dynamically allocate memory for n integers

int \*array = (int \*)malloc(n \* sizeof(int));

if (array == NULL) {

printf("Memory allocation failed.");

return 1;

}

// Accept data from the user

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

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

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

}

// Count non-zero elements and calculate sum

int count = 0, sum = 0;

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

if (array[i] != 0) {

count++;

sum += array[i];

}

}

// Allocate memory for non-zero elements array

int \*nonZeroArray = (int \*)malloc(count \* sizeof(int));

if (nonZeroArray == NULL) {

printf("Memory allocation failed.");

free(array);

return 1;

}

// Copy non-zero elements to the new array

int j = 0;

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

if (array[i] != 0) {

nonZeroArray[j++] = array[i];

}

}

// Calculate average

double average = (double)sum / count;

// Display non-zero elements, sum, and average

printf("Non-zero elements:\n");

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

printf("%d\n", nonZeroArray[i]);

}

printf("Sum of non-zero elements: %d\n", sum);

printf("Average of non-zero elements: %.2f\n", average);

// Free dynamically allocated memory

free(array);

free(nonZeroArray);

return 0;

}

Output:-

Enter the number of integers: 4

Enter 4 integers:

0

0

1

2

Non-zero elements:

1

2

Sum of non-zero elements: 3

Average of non-zero elements: 1.50

Exercise:-05

Q1] Input:-

#include<stdio.h>

#include<string.h>

int main() {

char string1[50],string2[50],string3[50],string4[50],string5[50],string6[50],string7[50];

int choise;

while(choise != 5) {

printf("\nThe Menu:-\n 1. Length\n 2. Copy\n 3. Concatenation\n 4. Compare\n");

printf("Enter the number of you'r choice from the menu:- ");

scanf("%d",&choise);

getchar();

switch(choise) {

case 1:

printf("you have chosen the length option from the meniu:-\n");

printf("Enter the string to calculate the length of it:- ");

fgets(string1,50, stdin);

int length=strlen(string1);

printf("The length of the string is:-%d",length-1);

break;

case 2:

printf("you have chosen the Copy option from the meniu:-\n");

printf("Enter the string1 to copy it in other string2:- ");

fgets(string2,50, stdin);

printf("The copy of the string is:-%s",strcpy(string3,string2));

break;

case 3:

printf("you have chosen the Concatenation option from the meniu:-\n");

printf("Enter the string1 And string2 to Concatenation it with the string1:- ");

fgets(string4,50, stdin);

fgets(string5,50, stdin);

printf("The Concatenation of the string is:-%s",strcat(string4,string5));

break;

case 4:

printf("you have chosen the Compare option from the meniu:-\n");

printf("Enter the string1 And string2 to Compare both:- ");

fgets(string6,50, stdin);

fgets(string7,50, stdin);

int compare=strcmp(string6,string7);

printf("The Compare of the string is:-%d",compare);

break;

default:

printf("you didn't enter from the menu");

}

}

}

Output:-

The Menu:-

1. Length

2. Copy

3. Concatenation

4. Compare

Enter the number of you'r choice from the menu:- 1

you have chosen the length option from the meniu:-

Enter the string to calculate the length of it:- Cdt.Akash.Kalidas.Durane

The length of the string is:-24

The Menu:-

1. Length

2. Copy

3. Concatenation

4. Compare

Enter the number of you'r choice from the menu:- 2

you have chosen the Copy option from the meniu:-

Enter the string1 to copy it in other string2:- Cdt.

The copy of the string is:-Cdt.

The Menu:-

1. Length

2. Copy

3. Concatenation

4. Compare

Enter the number of you'r choice from the menu:- 3

you have chosen the Concatenation option from the meniu:-

Enter the string1 And string2 to Concatenation it with the string1:- Cdt.Akash

.K.D

The Concatenation of the string is:-Cdt.Akash

.K.D

The Menu:-

1. Length

2. Copy

3. Concatenation

4. Compare

Enter the number of you'r choice from the menu:- 4

you have chosen the Compare option from the meniu:-

Enter the string1 And string2 to Compare both:- Cdt.Akash

Cdt.Akash

The Compare of the string is:-0

The Menu:-

1. Length

2. Copy

3. Concatenation

4. Compare

Enter the number of you'r choice from the menu:- 5

you didn't enter from the menu

Imp:- /\*

Your code seems mostly correct, but there's one issue related to the input reading. When you use **fgets()** to read strings, it reads the newline character (**\n**) left in the input buffer by the previous **scanf()** call. This leads to unexpected behavior in your program, causing it to skip input in some cases.

To fix this issue, you can add an additional **getchar()** call after **scanf("%d", &choise);** to consume the newline character.

\*/

Q2]Input:-

#include<stdio.h>

int position(char \*ptr,char ch);

int main() {

char ch,ptr[100];

printf("Ener the string:- ");

scanf("%s",ptr);

printf("Enter the character to be searched:- ");

scanf(" %c",&ch);

int positions=position(ptr,ch);

if(positions != -1) {

printf("The character %c is found at %d",ch,positions);

}

else {

printf("the character %c not found",ch);

}

}

int position(char \*ptr,char ch) {

int position = -1;

for(int i=0;ptr[i] != '\0';i++) {

if(ptr[i]==ch) {

return i+1;

}

}

return position;

}

Output:-

Ener the string:- Akashs

Enter the character to be searched:- s

The character s is found at 4

Imp:- /\*

1. When you use **scanf("%c", &ch);**, it reads the newline character (**\n**) left in the input buffer from the previous **scanf** for the string input. To fix this, you can add a space before **%c** like this: **scanf(" %c", &ch);**. This space tells **scanf** to ignore any leading whitespace characters, including the newline character.

\*/

Q3]Input:-

#include<stdio.h>

#include<string.h>

int is\_palidrome(char string[100]);

int main() {

char string[100];

printf("Enter the string to check its palidrome:- ");

scanf("%s",string);

if(is\_palidrome(string) == 1) {

printf("The entered string \"%s\"is a palidrome",string);

}

else{

printf("The entered string \"%s\"is not a palidrome",string);

}

}

int is\_palidrome(char string[100]) {

int start=0;

int n,end;

n=strlen(string);

end=n-1;

while(start<end) {

if(string[start] != string[end]) {

return 0;

}

start++;

end--;

}

return 1;

}

Output:-

Enter the string to check its palidrome:- maam

The entered string "maam"is a palidrome

Q4]input:-

#include<stdio.h>

#include<ctype.h>

#include<string.h>

void alter\_sentence(char sentence[100]);

int main() {

char sentence[100];

printf("Enter the sentence to alter it:- ");

fgets(sentence,100,stdin);

printf("the altered sentence is:- ");

alter\_sentence(sentence);

}

void alter\_sentence(char sentence[100]) {

for(int i=0;sentence[i] != '\0';i++) {

if(sentence[i]==' ') {

printf("\*");

}

else if (isalpha(sentence[i])) {

if(isupper(sentence[i])) {

printf("%c",tolower(sentence[i]));

}

else {

printf("%c",toupper(sentence[i]));

}

}

else if(isdigit(sentence[i])) {

printf("?");

}

else {

printf("%c",sentence[i]);

}

}

}

Output:-

Enter the sentence to alter it:- Cdt.Akash.Kalidas.Durane 232005

the altered sentence is:- cDT.aKASH.kALIDAS.dURANE\*??????

Exercise: - 6

Set A: -

Q1] #include <stdio.h>

#include <string.h>

#include <ctype.h>

void getStringFromUser(char \*str) {

printf("Enter a string: ");

fgets(str, 100, stdin);

}

int getStringLength(char \*str) {

int length = 0;

while (\*str != '\n') {

length++;

str++;

}

return length;

}

void reverseString(char \*str) {

int i, j;

char temp;

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

temp = \*(str + i);

\*(str + i) = \*(str + j);

\*(str + j) = temp;

}

}

void convertToUppercase(char \*str) {

while (\*str != '\0') {

\*str = toupper(\*str);

str++;

}

}

void convertToLowercase(char \*str) {

while (\*str != '\0') {

\*str = tolower(\*str);

str++;

}

}

int main() {

char str[100];

int choice;

do {

printf("\nString Operations Menu:\n");

printf("1. Get String Length\n");

printf("2. Reverse String\n");

printf("3. Convert to Uppercase\n");

printf("4. Convert to Lowercase\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Consume newline character

switch (choice) {

case 1:

getStringFromUser(str);

printf("Length of the string: %d\n", getStringLength(str));

break;

case 2:

getStringFromUser(str);

reverseString(str);

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

break;

case 3:

getStringFromUser(str);

convertToUppercase(str);

printf("String in uppercase: %s\n", str);

break;

case 4:

getStringFromUser(str);

convertToLowercase(str);

printf("String in lowercase: %s\n", str);

break;

case 5:

printf("Exiting program...\n");

break;

default:

printf("Invalid choice. Try again.\n");

}

} while (choice != 5);

return 0;

}

/\*

Important point to remember: -

The fgets function, when used to read input from the user, includes the newline character ('\n') in the string it reads, provided there's enough space in the buffer. This means that when the user enters "Cdt. Akash.Kalidas.Durane" and presses Enter, the input string stored in memory would be "Cdt. Akash.Kalidas.Durane\n" where \n represents the newline character.

int getStringLength(char \*str) {

int length = 0;

while (\*str != '\0') {

length++;

str++;

}

return length;

}

The loop increments the length count for each character until it encounters the null terminator ('\0'), which signifies the end of the string. Since fgets includes the newline character in the string it reads, the function counts it as well, resulting in a length of 26 instead of 25.

To rectify this and exclude the newline character from the count, we need to modify the function to terminate the counting when encountering either the newline character ('\n') or the null terminator ('\0'). Let's correct the function:

int getStringLength(char \*str) {

int length = 0;

while (\*str != '\0' && \*str != '\n') {

length++;

str++;

}

return length;

}

With this modification, the function will stop counting characters when it encounters either the newline character or the null terminator, thereby excluding the newline character from the length count.

\*/

OR

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int lentgthstr(char \*str);

char \*reversestr(char \*str);

char \*toupperstr(char \*str);

char \*tolowerstr(char \*str);

int main() {

char str[100];

int choice;

do {

printf("\nString Operations Menu:\n");

printf("1. Get String Length\n");

printf("2. Reverse String\n");

printf("3. Convert to Uppercase\n");

printf("4. Convert to Lowercase\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Consume newline character

switch(choice) {

case 1:

printf("Enter the string:- ");

fgets(str,100,stdin);

int length=lentgthstr(str);

printf("The length of the string is:- %d",length);

break;

case 2:

printf("Enter the string:- ");

fgets(str,100,stdin);

printf("The reverse of the string is:- %s",reversestr(str));

break;

case 3:

printf("Enter the string:- ");

fgets(str,100,stdin);

printf("The toupper of the string is:- %s",toupperstr(str));

break;

case 4:

printf("Enter the string:- ");

fgets(str,100,stdin);

printf("The tolower of the string is:- %s",tolowerstr(str));

break;

default: printf("Invalid input");

}

}

while(choice != 5);

}

int lentgthstr(char \*str) {

int length=0;

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

length++;

}

return length;

}

char \*reversestr(char \*str) {

int l,r;

l=0;

r=strlen(str)-1;

while(l<r) {

char temp= str[l];

str[l]=str[r];

str[r]=temp;

l++;

r--;

}

return str;

}

char \*toupperstr(char \*str) {

for(int i=0;str[i] != '\0';i++) {

if(islower(str[i])) {

str[i]=toupper(str[i]);

}

}

return str;

}

char \*tolowerstr(char \*str) {

for(int i=0;str[i] != '\0';i++) {

if(isupper(str[i])) {

str[i]=tolower(str[i]);

}

}

return str;

}

Output: -

String Operations Menu:

1. Get String Length

2. Reverse String

3. Convert to Uppercase

4. Convert to Lowercase

5. Exit

Enter your choice: 2

Enter the string:- Cdt.Akash.Kalidas.Durane

The reverse of the string is:-

enaruD.sadilaK.hsakA.tdC

String Operations Menu:

1. Get String Length

2. Reverse String

3. Convert to Uppercase

4. Convert to Lowercase

5. Exit

Enter your choice: 3

Enter the string:- Cdt. Akash.Kalidas.Durane

The toupper of the string is:- CDT. AKASH.KALIDAS.DURANE

String Operations Menu:

1. Get String Length

2. Reverse String

3. Convert to Uppercase

4. Convert to Lowercase

5. Exit

Enter your choice: 4

Enter the string:- Cdt. Akash.Kalidas.Durane

The tolower of the string is:- cdt. akash.kalidas.durane

String Operations Menu:

1. Get String Length

2. Reverse String

3. Convert to Uppercase

4. Convert to Lowercase

5. Exit

Enter your choice: 1

Enter the string:- Cdt. Akash.Kalidas.Durane

The length of the string is:- 25

String Operations Menu:

1. Get String Length

2. Reverse String

3. Convert to Uppercase

4. Convert to Lowercase

5. Exit

Enter your choice: 5

Invalid input

Q2] Input: -

#include <stdio.h>

#include <string.h>

void getStringsFromUser(char \*str1, char \*str2) {

printf("Enter the first string: ");

fgets(str1, 100, stdin);

str1[strcspn(str1, "\n")] = '\0'; // Remove newline character

printf("Enter the second string: ");

fgets(str2, 100, stdin);

str2[strcspn(str2, "\n")] = '\0'; // Remove newline character

}

char\* copyString(char \*dest, char \*src) {

return strcpy(dest,src);

}

char\* concatenateStrings(char \*str, char \*str3) {

return strcat(str,str3);

}

int compareStrings(char \*str1, char \*str2) {

return strcmp(str1,str2);

}

int main() {

char str1[100], str2[100], result[200],str[100],str3[100];

int choice;

do {

printf("\nString Operations Menu:\n");

printf("1. Copy String\n");

printf("2. Concatenate Strings\n");

printf("3. Compare Strings\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Consume newline character

switch (choice) {

case 1:

getStringsFromUser(str1, str2);

printf("Original string: %s\n", str1);

printf("Copied string: %s\n", copyString(result, str1));

break;

case 2:

getStringsFromUser(str, str3);

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

printf("Second string: %s\n", str3);

printf("Concatenated string: %s\n", concatenateStrings(result, str2));

break;

case 3:

getStringsFromUser(str1, str2);

printf("First string: %s\n", str1);

printf("Second string: %s\n", str2);

int cmp = compareStrings(str1, str2);

if (cmp < 0) {

printf("First string is smaller\n");

} else if (cmp > 0) {

printf("Second string is smaller\n");

} else {

printf("Strings are equal\n");

}

break;

case 4:

printf("Exiting program...\n");

break;

default:

printf("Invalid choice. Try again.\n");

}

} while (choice != 4);

return 0;

}

Output: -

String Operations Menu:

1. Copy String

2. Concatenate Strings

3. Compare Strings

4. Exit

Enter your choice: 1

Enter the first string: Cdt.Akash

Enter the second string: .Kalidas.Durane

Original string: Cdt.Akash

Copied string: Cdt.Akash

String Operations Menu:

1. Copy String

2. Concatenate Strings

3. Compare Strings

4. Exit

Enter your choice: 2

Enter the first string: Cdt. Akash.Kalidas

Enter the second string: .Durane

First string: Cdt. Akash.Kalidas

Second string: .Durane

Concatenated string: Cdt.Akash.Kalidas.Durane

String Operations Menu:

1. Copy String

2. Concatenate Strings

3. Compare Strings

4. Exit

Enter your choice: 3

Enter the first string: Akash

Enter the second string: Akash

First string: Akash

Second string: Akash

Strings are equal

String Operations Menu:

1. Copy String

2. Concatenate Strings

3. Compare Strings

4. Exit

Enter your choice: 4

Exiting program...

Exercise: -07

Set A: -

Q1] Input: -

#include <stdio.h>

#include <string.h>

#define MAX\_STUDENTS 100

struct student {

int rollNumber;

char name[50];

float marks[3];

float percentage;

};

void calculatePercentage(struct student\* students, int numStudents) {

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

float totalMarks = 0;

for (int j = 0; j < 3; j++) {

totalMarks += students[i].marks[j];

}

students[i].percentage = (totalMarks / 300) \* 100;

}

}

void acceptStudentDetails(struct student \*students, int \*numStudents) {

printf("Enter the number of students: ");

scanf("%d", numStudents);

for (int i = 0; i < \*numStudents; i++) {

printf("\nEnter details for student %d:\n", i + 1);

printf("Roll Number: ");

scanf("%d", &students[i].rollNumber);

printf("Name: ");

scanf(" %[^\n]s", students[i].name);

for (int j = 0; j < 3; j++) {

printf("Marks for subject %d: ", j + 1);

scanf("%f", &students[i].marks[j]);

}

}

calculatePercentage(students, \*numStudents);

}

void searchByRollNumber(struct student\* students, int numStudents) {

int rollNumber;

printf("Enter the roll number to search: ");

scanf("%d", &rollNumber);

int found = 0;

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

if (students[i].rollNumber == rollNumber) {

printf("\nStudent Details:\n");

printf("Roll Number: %d\n", students[i].rollNumber);

printf("Name: %s\n", students[i].name);

printf("Marks: %.2f, %.2f, %.2f\n", students[i].marks[0], students[i].marks[1], students[i].marks[2]);

printf("Percentage: %.2f%%\n", students[i].percentage);

found = 1;

break;

}

}

if (!found) {

printf("Student with roll number %d not found.\n", rollNumber);

}

}

void modifyStudentName(struct student\* students, int numStudents) {

int rollNumber;

printf("Enter the roll number of the student whose name you want to modify: ");

scanf("%d", &rollNumber);

int found = 0;

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

if (students[i].rollNumber == rollNumber) {

printf("Current name: %s\n", students[i].name);

printf("Enter the new name: ");

scanf(" %[^\n]s", students[i].name);

found = 1;

break;

}

}

if (!found) {

printf("Student with roll number %d not found.\n", rollNumber);

}

}

void displayAllStudents(struct student\* students, int numStudents) {

printf("\nStudent Details:\n");

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

printf("Roll Number: %d\n", students[i].rollNumber);

printf("Name: %s\n", students[i].name);

printf("Marks: %.2f, %.2f, %.2f\n", students[i].marks[0], students[i].marks[1], students[i].marks[2]);

printf("Percentage: %.2f%%\n\n", students[i].percentage);

}

}

void displayStudentsWithPercentageGreaterThan75(struct student\* students, int numStudents) {

int found = 0;

printf("\nStudents with percentage greater than 75%%:\n");

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

if (students[i].percentage > 75) {

printf("Roll Number: %d\n", students[i].rollNumber);

printf("Name: %s\n", students[i].name);

printf("Marks: %.2f, %.2f, %.2f\n", students[i].marks[0], students[i].marks[1], students[i].marks[2]);

printf("Percentage: %.2f%%\n\n", students[i].percentage);

found = 1;

}

}

if (!found) {

printf("No students found with percentage greater than 75%%.\n");

}

}

void displayStudentWithMaxPercentage(struct student\* students, int numStudents) {

int maxIndex = 0;

float maxPercentage = students[0].percentage;

for (int i = 1; i < numStudents; i++) {

if (students[i].percentage > maxPercentage) {

maxPercentage = students[i].percentage;

maxIndex = i;

}

}

printf("\nStudent with maximum percentage:\n");

printf("Roll Number: %d\n", students[maxIndex].rollNumber);

printf("Name: %s\n", students[maxIndex].name);

printf("Marks: %.2f, %.2f, %.2f\n", students[maxIndex].marks[0], students[maxIndex].marks[1], students[maxIndex].marks[2]);

printf("Percentage: %.2f%%\n", students[maxIndex].percentage);

}

int main() {

struct student students[MAX\_STUDENTS];

int numStudents = 0;

int choice;

do {

printf("\nStudent Management System\n");

printf("1. Accept student details\n");

printf("2. Search by roll number\n");

printf("3. Modify student name\n");

printf("4. Display all student details\n");

printf("5. Display students with percentage > 75\n");

printf("6. Display student with maximum percentage\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

acceptStudentDetails(students, &numStudents);

break;

case 2:

searchByRollNumber(students, numStudents);

break;

case 3:

modifyStudentName(students, numStudents);

break;

case 4:

displayAllStudents(students, numStudents);

break;

case 5:

displayStudentsWithPercentageGreaterThan75(students, numStudents);

break;

case 6:

displayStudentWithMaxPercentage(students, numStudents);

break;

case 7:

printf("Exiting program...\n");

break;

default:

printf("Invalid choice. Try again.\n");

}

} while (choice != 7);

return 0;

}

/\*

It seems like you've provided a format specifier %[^\n]s. This format specifier is used in scanf() function in C to read a string until a newline character ('\n') is encountered, excluding the newline character itself.

\*/