**Assignment -2**

**Ishita Yadav**

**24csu084**

Q1. WAP to increase every student mark by 5 & then print the updated array.

Sol:

#include <stdio.h>

int main() {

int marks[5],i;

for(i=0;i<=4;i=i+1) {

printf("Enter marks of student-%d :",i+1);

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

}

printf("\n# Grade list of student's as per marks\n");

for(i=0;i<=4;i=i+1) {

if (marks[i]>=75)

printf("Student-%d has achieved Grade-A\n",i+1);

else if (marks[i]>=60 && marks[i]<75)

printf("Student-%d has achieved Grade-B\n",i+1);

else if (marks[i]>=40 && marks[i]<60)

printf("Student-%d has achieved Grade-C\n",i+1);

else if (marks[i]>=0 && marks[i]<40)

printf("Student-%d has achieved Grade-D\n",i+1);

else

printf("NOTE--Enter marks on scale of 0-100.\n");

}

return 0;

}

-----------------------------------------------------------------------------

Q2. WAP to print grade of students as per their marks given in an array. (>=75-- A  
grade, 74 to 60--B Grade, 59 to 40--C grade below 40--D grade).

Sol:

#include <stdio.h>

int main() {

int marks[5],i;

for(i=0;i<=4;i=i+1) {

printf("Enter marks of student-%d :",i+1);

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

}

printf("\n# Grade list of student's as per marks\n");

for(i=0;i<=4;i=i+1) {

if (marks[i]>=75)

printf("Student-%d has achieved Grade-A\n",i+1);

else if (marks[i]>=60 && marks[i]<75)

printf("Student-%d has achieved Grade-B\n",i+1);

else if (marks[i]>=40 && marks[i]<60)

printf("Student-%d has achieved Grade-C\n",i+1);

else if (marks[i]>=0 && marks[i]<40)

printf("Student-%d has achieved Grade-D\n",i+1);

else

printf("NOTE--Enter marks on scale of 0-100.\n");

}

return 0;

}

Q3. WAP to find who scored first “99” in an array marks.

Sol:

#include <stdio.h>

int main() {

int marks[5],i;

for(i=0;i<=4;i=i+1) {

printf("Enter marks of student-%d :",i+1);

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

}

for(i=0;i<=4;i=i+1) {

if (marks[i]==99){

printf("Student-%d has scored first 99 marks",i+1);

break;

}

}

return 0;

}

Q4. WAP to find Who & how many students have scored 99 in an array Marks.

Sol:

#include <stdio.h>

int main() {

int marks[5],i,sum=0;

for(i=0;i<=4;i=i+1) {

printf("Enter marks of student-%d :",i+1);

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

}

for(i=0,sum;i<=4;i=i+1) {

if (marks[i]==99){

printf("\nStudent-%d has scored 99 marks",i+1);

sum=sum+1;

}

}

printf("\n\n%d student's has scored 99 marks",sum);

return 0;

}

Q5. WAP to find sum of all scores in Marks array

Sol:

#include <stdio.h>

int main() {

int marks[5],i,sum=0;

for(i=0;i<=4;i=i+1) {

printf("Enter marks of student-%d :",i+1);

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

}

for(i=0,sum;i<=4;i=i+1) {

sum=sum+marks[i];

}

printf("\nSum of mark's of all student's is %d",sum);

}

-------------------------------------------------------------------------------

Q6. WAP to find average score of the Marks array.

Sol:

#include <stdio.h>

int average (int sum) {

int a;

a=sum/5;

return a;

};

int main() {

int marks[5],i,sum=0;

for(i=0;i<=4;i=i+1) {

printf("Enter marks of student-%d :",i+1);

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

}

for(i=0,sum;i<=4;i=i+1) {

sum=sum+marks[i];

}

printf("\nSum of mark's of all student's is %d\n",sum);

printf("Average of total marks is %d",average(sum));

return 0;

}

------------------------------------------------------------------------------

Q7. WAP to check whether score is even or odd in an array.

Sol:

#include <stdio.h>

int main() {

int i,score[5];

for(i=0;i<=4;i=i+1) {

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

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

}

printf("\n");

for (i=0;i<=4;i=i+1) {

if (score[i]%2==0)

printf("Score-%d (%d) is even\n",i+1,score[i]);

else

printf("Score-%d (%d) is odd\n",i+1,score[i]);

}

return 0;

}

--------------------------------------------------------------------------------

Q8. WAP to find maximum & minimum score in the Marks array

Sol:

#include <stdio.h>

int main() {

int n, i;

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

scanf("%d", &n);

int marks[n];

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

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

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

}

int max = marks[0];

int min = marks[0];

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

if(marks[i] > max) {

max = marks[i];

}

if(marks[i] < min) {

min = marks[i];

}

}

printf("Maximum score: %d\n", max);

printf("Minimum score: %d\n", min);

return 0;

}

Q9. WAP to find a peak element which is not smaller than its neighbors.

Sol:

#include <stdio.h>

int findPeak(int arr[], int n) {

if (n == 1 || arr[0] >= arr[1]) return 0;

if (arr[n - 1] >= arr[n - 2]) return n - 1;

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

if (arr[i] >= arr[i - 1] && arr[i] >= arr[i + 1]) {

return i;

}

}

return -1;

}

int main() {

int n;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

int peakIndex = findPeak(arr, n);

if (peakIndex != -1) {

printf("A peak element is at index %d\n", peakIndex);

} else {

printf("No peak element found\n");

}

return 0;

}

Q10. WAP to count prime numbers in an array.

Sol:

#include <stdio.h>

int isPrime(int num) {

if (num <= 1) return 0;

for (int i = 2; i <= num / 2; i++) {

if (num % i == 0) return 0;

}

return 1;

}

int main() {

int n, count = 0;

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

scanf("%d", &n);

int arr[n];

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

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

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

}

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

if (isPrime(arr[i])) {

count++;

}

}

printf("The number of prime numbers in the array is: %d\n", count);

return 0;

}

------------------------------------------------------------------------------

Q11. WAP to implement Insert -Front, any position in between & end in an array. Print the array before insert & after insert.

Sol:

#include <stdio.h>

#define MAX\_SIZE 100

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

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

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

}

printf("\n");

}

void insertAtFront(int arr[], int \*size, int value) {

if (\*size >= MAX\_SIZE) {

printf("Array is full\n");

return;

}

for (int i = \*size; i > 0; i--) {

arr[i] = arr[i - 1];

}

arr[0] = value;

(\*size)++;

}

void insertAtEnd(int arr[], int \*size, int value) {

if (\*size >= MAX\_SIZE) {

printf("Array is full\n");

return;

}

arr[\*size] = value;

(\*size)++;

}

void insertAtPosition(int arr[], int \*size, int value, int position) {

if (\*size >= MAX\_SIZE) {

printf("Array is full\n");

return;

}

if (position < 0 || position > \*size) {

printf("Invalid position\n");

return;

}

for (int i = \*size; i > position; i--) {

arr[i] = arr[i - 1];

}

arr[position] = value;

(\*size)++;

}

int main() {

int arr[MAX\_SIZE];

int size, value, position;

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

scanf("%d", &size);

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

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

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

}

printf("Array before insertion:\n");

printArray(arr, size);

printf("Enter the value to insert at the front: ");

scanf("%d", &value);

insertAtFront(arr, &size, value);

printf("Array after inserting at the front:\n");

printArray(arr, size);

printf("Enter the value to insert at the end: ");

scanf("%d", &value);

insertAtEnd(arr, &size, value);

printf("Array after inserting at the end:\n");

printArray(arr, size);

printf("Enter the value to insert and the position: ");

scanf("%d %d", &value, &position);

insertAtPosition(arr, &size, value, position);

printf("Array after inserting at position %d:\n", position);

printArray(arr, size);

return 0;

}

Q12. WAP to implement delete-Front, any position in between & end in an array. Print the array before delete & after delete

Sol:

#include <stdio.h>

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

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

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

}

printf("\n");

}

void deleteFront(int arr[], int \*size) {

if (\*size <= 0) {

printf("Array is empty.\n");

return;

}

for (int i = 0; i < \*size - 1; i++) {

arr[i] = arr[i + 1];

}

(\*size)--;

}

void deleteEnd(int arr[], int \*size) {

if (\*size <= 0) {

printf("Array is empty.\n");

return;

}

(\*size)--;

}

void deleteAtPosition(int arr[], int \*size, int position) {

if (position < 0 || position >= \*size) {

printf("Invalid position.\n");

return;

}

for (int i = position; i < \*size - 1; i++) {

arr[i] = arr[i + 1];

}

(\*size)--;

}

int main() {

int arr[100], size, choice, position;

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

scanf("%d", &size);

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

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

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

}

printf("Array before deletion: ");

printArray(arr, size);

printf("Choose deletion method:\n");

printf("1. Delete from front\n");

printf("2. Delete from end\n");

printf("3. Delete from a specific position\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

deleteFront(arr, &size);

break;

case 2:

deleteEnd(arr, &size);

break;

case 3:

printf("Enter the position to delete: ");

scanf("%d", &position);

deleteAtPosition(arr, &size, position);

break;

default:

printf("Invalid choice.\n");

return 1;

}

printf("Array after deletion: ");

printArray(arr, size);

return 0;

}

Q13. Given an array, the task is to cyclically rotate the array clockwise by one time.  
Examples:  
Input: arr[] = {1, 2, 3, 4, 5}  
Output: arr[] = {5, 1, 2, 3, 4}  
Input: arr[] = {2, 3, 4, 5, 1}  
Output: {1, 2, 3, 4, 5}

Sol:

#include <stdio.h>

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

int temp = arr[n-1]; // Store the last element in a temporary variable

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

arr[i] = arr[i-1]; // Shift elements to the right

}

arr[0] = temp; // Place the last element at the first position

}

int main() {

int arr[] = {1, 2, 3, 4, 5}; // Example array

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

printf("Original array: ");

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

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

}

rotateByOne(arr, n);

printf("\nRotated array: ");

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

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

}

return 0;

}

Q14. Given an array of n integers. The task is to print the duplicates in the given array.  
If there are no duplicates then print -1.  
Examples:  
Input: {2, 10,10, 100, 2, 10, 11,2,11,2}  
Output: 2 10 11  
Input: {5, 40, 1, 40, 100000, 1, 5, 1}  
Output: 5 40 1

Sol:

#include <stdio.h>

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

int foundDuplicate = 0;

int count[n];

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

count[i] = 0;

}

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

count[arr[i]]++;

}

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

if (count[arr[i]] > 1) {

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

count[arr[i]] = 0;

foundDuplicate = 1;

}

}

if (!foundDuplicate) {

printf("-1");

}

}

int main() {

int arr[] = {1, 2, 3, 4, 2, 3, 5};

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

printDuplicates(arr, n);

return 0;

}