LAB ASSIGNMENT – 2

U24cs076

Rushang bagada

Q. 1: - Write a C program to create three signed integer arrays SubArray1,

SubArray2, and

MainArray of sizes 10, 10, and 20 respectively. With the appropriate error

message

handling, perform the following operations on the array with the help of the

menu.

After each operation, print the array elements.

a. Insert the elements in a user-given array and print the memory address of all

elements, the size of the array, and the number of elements in the array:

- at the end

- at the beginning

- at the user-given position

b. Modify the elements in a user-given array:

- at the end

- at the beginning

- at the user-given position

c. Delete the element from the array:

- from the end

- from the beginning and

- from the user-given position

d. Reverse the elements of the user-given array:

- use a temporary array to hold the result

- do not create another array to hold the result

e. Copy the sequence of the user-given subarray to Main Array.

f. Merge two arrays of the same size sorted in descending order.

g. Display the sum of all array elements:

• without recursion

• using recursion

h. Search the array for the user-given value from the user-given array.

i. From the MainArray, separate sub-arrays for odd and even numbers and

display

them. (Assume 10 odd and 10 even numbers are provided in MainArray).

j. From the MainArray, separate sub-arrays for positive and negative numbers

and

display them. (Assume 10 positive and 10 negative numbers are provided in

MainArray).

Code :-

#include<stdio.h>

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

int i;

for(i=0;i<size;i++)printf("%d ",arr[i]);

printf("\n%d\n",size);

}

void insertAtBeginning(int arr[],int \*size,int capacity,int val){

if(\*size==capacity){

printf("Array is Full.\n");return;

}

int i;

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

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

}

arr[0]=val;

(\*size)++;

}

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

if(\*size==capacity){

printf("Array is Full.\n");return;

}

arr[\*size]=val;

(\*size)++;

}

void insertAtPos(int arr[],int \*size,int capacity,int pos,int val){

if(\*size==capacity){

printf("Array is Full.\n");return;

}

int i;

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

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

}

arr[pos]=val;

(\*size)++;

}

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

if(\*size==0){

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

return;

}

arr[0]=val;

}

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

if(\*size==0){

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

return;

}

arr[\*size-1]=val;

}

void modifyAtPos(int arr[],int pos,int val){

arr[pos]=val;

}

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

if(\*size==0){

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

}

int i;

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

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

}

(\*size)--;

}

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

if(\*size==0){

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

}

(\*size)--;

}

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

if(\*size==0){

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

}

int i;

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

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

}

(\*size)--;

}

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

if(size==0){

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

}

int i;

for(i=0;i<size/2;i++){

int t=arr[i];arr[i]=arr[size-1-i];arr[size-1-i]=t;

}

}

void Merge(int arr[],int n1,int brr[],int n2,int ans[],int \*sizeM){

int i=0,j=0,k=0;

while(i<n1 && j<n2){

if(arr[i]>=brr[j])ans[k++]=arr[i++];

else ans[k++]=brr[j++];

}

if(i==n1)while(j<n2)ans[k++]=brr[j++];

if(j==n2)while(i<n1)ans[k++]=arr[i++];

\*sizeM=k;

}

void copy(int a1[],int \*size1,int a2[],int size2){

int i;

for(i=0;i<size2;i++){

a1[i]=a2[i];

}

\*size1=size2;

}

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

int sum=0,i;

for(i=0;i<size;i++)sum+=arr[i];

printf("%d\n",sum);

}

int displaySumRec(int arr[],int size,int sum,int i){

if(i==size)return sum;

return displaySumRec(arr,size,sum+arr[i],i+1);

}

void find(int arr[],int size,int val){

int i;

for(int i=0;i<size;i++)if(arr[i]==val)break;

if(i==size)printf("Element not found.\n");

else{

printf("Element found at intdex %d",i);

}

}

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

int i;

printf("Odd numbers are : ");

for(i=0;i<size;i++)if(arr[i]%2!=0)printf("%d ",arr[i]);

printf("\n");

printf("Even numbers are : ");

for(i=0;i<size;i++)if(arr[i]%2==0)printf("%d ",arr[i]);

printf("\n");

}

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

int i;

printf("Positive numbers are : ");

for(i=0;i<size;i++)if(arr[i]>=0)printf("%d ",arr[i]);

printf("\n");

printf("Negative numbers are : ");

for(i=0;i<size;i++)if(arr[i]<0)printf("%d ",arr[i]);

printf("\n");

}

int main(){

int SubArray1[10],size1=0;

int SubArray2[10],size2=0;

int MainArray[20],sizeM=0;

while(1){

printf("Would you like to continue(Y/N)");char ch;scanf("%c",&ch);

if(ch!='Y')break;

printf("To insert, click 1\nTo modify,click 2\nTo delete, click 3\nTo reverse

elements,click 4\nTo copy elements of SubArray to MainArray, click 5\nTo

merge two arrays of same size sorted in descending order, click 6\nTo display

sum of elements, click 7\nTo find element, click 8\nTo Display odd and even

elements of MainArray, click 9\nTo Display positive and negative elements of

MainArray, click 10\n");

int root;

scanf("%d",&root);

switch(root){

//INSERT

case 1:{

printf("Enter val you want to insert:");int val;scanf("%d",&val);

printf("To insert at beginning : click 1\nTo insert at end, click 2\nTo

insert at any index, click 3\n");

int child;scanf("%d",&child);

printf("To insert in SubArray1 : click 1\nTo insert in SubArray2, click

2\nTo insert in MainArray, click 3\n");

int grandchild;scanf("%d",&grandchild);

//TYPE OF INSERT

switch(child){

case 1:{

//IN WHICH ARRAY

switch(grandchild){

case 1:{

insertAtBeginning(SubArray1,&size1,10,val);

display(SubArray1,size1);

break;

}

case 2:{

insertAtBeginning(SubArray2,&size2,10,val);

display(SubArray2,size2);

break;

}

case 3:{

insertAtBeginning(MainArray,&sizeM,20,val);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

break;

}

case 2:{

//IN WHICH ARRAY

switch(grandchild){

case 1:{

insertAtEnd(SubArray1,&size1,10,val);

display(SubArray1,size1);

break;

}

case 2:{

insertAtEnd(SubArray2,&size2,10,val);

display(SubArray2,size2);

break;

}

case 3:{

insertAtEnd(MainArray,&sizeM,20,val);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

break;

}

case 3:{

int pos;printf("Enter index at which you want to

insert:");scanf("%d",&pos);

//IN WHICH ARRAY

switch(grandchild){

case 1:{

if(pos>=0 && pos<=size1){

insertAtPos(SubArray1,&size1,10,pos,val);

display(SubArray1,size1);

}

else printf("Invalid index!\n");

break;

}

case 2:{

if(pos>=0 && pos<=size2){

insertAtPos(SubArray2,&size2,10,pos,val);

display(SubArray2,size2);

}

else printf("Invalid index!\n");

break;

}

case 3:{

if(pos>=0 && pos<=sizeM){

insertAtPos(MainArray,&sizeM,20,pos,val);

display(MainArray,sizeM);

}

else printf("Invalid index!\n");

break;

}

default : printf("Invalid Input");

}

break;

}

default : printf("Invalid input!\n");

}

break;

}

//MODIFY

case 2:{

printf("Enter updated value:");int val;scanf("%d",&val);

printf("To modify at beginning, click 1\nTo modify at end, click 2\nTo

modify at any index, click 3\n");

int child;scanf("%d",&child);

printf("To modify in SubArray1 : click 1\nTo modify in SubArray2, click

2\nTo modify in MainArray, click 3\n");

int grandchild;scanf("%d",&grandchild);

//TYPE OF MODIFY

switch(child){

case 1:{

//IN WHICH ARRAY

switch(grandchild){

case 1:{

modifyAtBeginning(SubArray1,&size1,val);

display(SubArray1,size1);

break;

}

case 2:{

modifyAtBeginning(SubArray2,&size2,val);

display(SubArray2,size2);

break;

}

case 3:{

modifyAtBeginning(MainArray,&sizeM,val);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

break;

}

case 2:{

//IN WHICH ARRAY

switch(grandchild){

case 1:{

modifyAtEnd(SubArray1,&size1,val);

display(SubArray1,size1);

break;

}

case 2:{

modifyAtEnd(SubArray2,&size2,val);

display(SubArray2,size2);

break;

}

case 3:{

modifyAtEnd(MainArray,&sizeM,val);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

break;

}

case 3:{

int pos;printf("Enter index at which you want to

insert:");scanf("%d",&pos);

//IN WHICH ARRAY

switch(grandchild){

case 1:{

if(pos>=0 && pos<size1){

modifyAtPos(SubArray1,pos,val);

display(SubArray1,size1);

}

else printf("Invalid index!\n");

break;

}

case 2:{

if(pos>=0 && pos<size2){

modifyAtPos(SubArray2,pos,val);

display(SubArray2,size2);

}

else printf("Invalid index!\n");

break;

}

case 3:{

if(pos>=0 && pos<sizeM){

modifyAtPos(MainArray,pos,val);

display(MainArray,sizeM);

}

else printf("Invalid index!\n");

break;

}

default : printf("Invalid Input");

}

break;

}

default : printf("Invalid input!\n");

}

break;

}

//DELETE

case 3:{

printf("To delete at beginning, click 1\nTo delete at end, click 2\nTo

delete at any index, click 3\n");

int child;scanf("%d",&child);

printf("To delete in SubArray1, click 1\nTo delete in SubArray2, click

2\nTo delete in MainArray, click 3\n");

int grandchild;scanf("%d",&grandchild);

//TYPE OF DELETE

switch(child){

case 1:{

//IN WHICH ARRAY

switch(grandchild){

case 1:{

deleteAtBeginning(SubArray1,&size1);

display(SubArray1,size1);

break;

}

case 2:{

deleteAtBeginning(SubArray2,&size2);

display(SubArray2,size2);

break;

}

case 3:{

deleteAtBeginning(MainArray,&sizeM);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

break;

}

case 2:{

//IN WHICH ARRAY

switch(grandchild){

case 1:{

deleteAtEnd(SubArray1,&size1);

display(SubArray1,size1);

break;

}

case 2:{

deleteAtEnd(SubArray2,&size2);

display(SubArray2,size2);

break;

}

case 3:{

deleteAtEnd(MainArray,&sizeM);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

break;

}

case 3:{

int pos;printf("Enter index at which you want to

delete:");scanf("%d",&pos);

//IN WHICH ARRAY

switch(grandchild){

case 1:{

if(pos>=0 && pos<size1){

deleteAtPos(SubArray1,&size1,pos);

display(SubArray1,size1);

}

else printf("Invalid index!\n");

break;

}

case 2:{

if(pos>=0 && pos<size2){

deleteAtPos(SubArray2,&size2,pos);

display(SubArray2,size2);

}

else printf("Invalid index!\n");

break;

}

case 3:{

if(pos>=0 && pos<sizeM){

deleteAtPos(MainArray,&sizeM,pos);

display(MainArray,sizeM);

}

else printf("Invalid index!\n");

break;

}

default : printf("Invalid Input");

}

break;

}

default : printf("Invalid input!\n");

}

break;

}

//REVERSE

case 4:{

//IN WHICH ARRAY

printf("To reverse in SubArray1, click 1\nTo reverse in SubArray2, click

2\nTo reverse in MainArray, click 3\n");

int child;scanf("%d",&child);

switch(child){

case 1:{

reverse(SubArray1,size1);

display(SubArray1,size1);

break;

}

case 2:{

reverse(SubArray2,size2);

display(SubArray2,size2);

break;

}

case 3:{

reverse(MainArray,sizeM);

display(MainArray,sizeM);

break;

}

default : printf("Invalid Input");

}

}

//COPY

case 5:{

printf("To copy SubArray1, click 1\nTo copy SubArray2, click 2\n");

int child;scanf("%d",&child);

switch(child){

case 1: {

copy(MainArray,&sizeM,SubArray1,size1);

display(MainArray,sizeM);

break;

}

case 2: {

copy(MainArray,&sizeM,SubArray2,size2);

display(MainArray,sizeM);

break;

}

default : printf("Invalid input!\n");

}

break;

}

//MERGE

case 6:{

if(size1==size2){

Merge(SubArray1,size1,SubArray2,size2,MainArray,&sizeM);

display(MainArray,sizeM);

}

else printf("Sizes of two arrays to be merged are unequal\n");

break;

}

//DISPLAY SUM

case 7:{

printf("To Display sum using recursion, click 1\nTo Display sum without

using recursion, click 2\n");

int child;scanf("%d",&child);

switch(child){

case 1:{

printf("To display sum of SubArray1, click 1\nTo display sum of

SubArray2, click 2\nTo display sum of MainArray, click 3\n");

int grandchild;scanf("%d",&grandchild);

int sum;

switch(grandchild){

case 1:{

sum=displaySumRec(SubArray1,size1,0,0);

break;

}

case 2:{

sum=displaySumRec(SubArray2,size2,0,0);

break;

}

case 3:{

sum=displaySumRec(MainArray,sizeM,0,0);

break;

}

default: printf("Wrong input!\n");

}

printf("%d\n",sum);

break;

}

case 2:{

printf("To display sum of SubArray1, click 1\nTo display sum of

SubArray2, click 2\nTo display sum of MainArray, click 3\n");

int grandchild;scanf("%d",&grandchild);

switch(grandchild){

case 1:{

displaySum(SubArray1,size1);

break;

}

case 2:{

displaySum(SubArray2,size2);

break;

}

case 3:{

displaySum(MainArray,sizeM);

break;

}

default: printf("Wrong input!\n");

}

}

}

break;

}

//FIND

case 8:{

printf("Enter the value you want to find:");

int val;scanf("%d",&val);

printf("To find element in SubArray1, click 1\nTo find element in

SubArray2, click 2\nTo find element in MainArray, click 3\n");

int child;scanf("%d",&child);

switch(child){

case 1:{

find(SubArray1,size1,val);

break;

}

case 2:{

find(SubArray2,size2,val);

break;

}

case 3:{

find(MainArray,sizeM,val);

break;

}

default: printf("Wrong input!\n");

}

break;

}

//DIDPLAY ODD AND EVEN

case 9:{

displayOddEven(MainArray,sizeM);

break;

}

//DIDPLAY POSITIVE AND NEGATIVE

case 10:{

displayPosNeg(MainArray,sizeM);

break;

}

default : printf("Wronng input!\n");

}

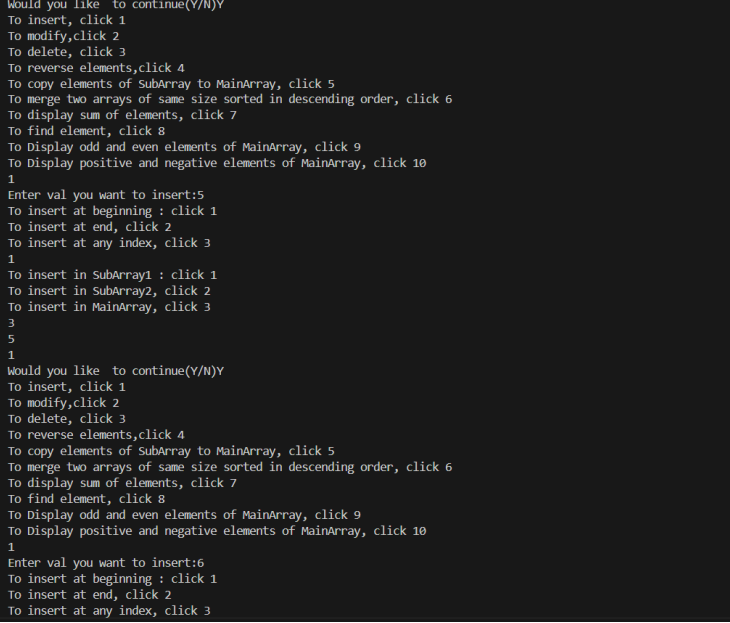
fflush(stdin);

}

return 0;

}

OUTPUT:



Q.2: - Calendar. Repeat Exercise 1.33 to produce a calendar for a given month

and year. Use

arrays to store the names of the days of the week, the names of the months,

and the

number of days in a month.

Code:

#include <stdio.h>

void day(int n){

n=n%7;

switch(n){

case 1:{printf("Wednesday\n");break;}

case 2:{printf("Thursday\n");break;}

case 3:{printf("Friday\n");break;}

case 4:{printf("Saturday\n");break;}

case 5:{printf("Sunday\n");break;}

case 6:{printf("Monday\n");break;}

case 0:{printf("Tuesday\n");break;}

}

}

int main(){

printf("Input month number: ");int m;scanf("%d",&m);

if(m>12 && m<1)printf("Invalid month!");

else{

int sum=0;

switch(m-1){

case 12:sum+=31;

case 11:sum+=30;

case 10:sum+=31;

case 9:sum+=30;

case 8:sum+=31;

case 7:sum+=31;

case 6:sum+=30;

case 5:sum+=31;

case 4:sum+=30;

case 3:sum+=31;

case 2:sum+=28;

case 1:sum+=31;

}

int size;

switch(m){

case 1:{size=31;break;}

case 2:{size=28;break;}

case 3:{size=31;break;}

case 4:{size=30;break;}

case 5:{size=31;break;}

case 6:{size=30;break;}

case 7:{size=31;break;}

case 8:{size=31;break;}

case 9:{size=30;break;}

case 10:{size=31;break;}

case 11:{size=30;break;}

case 12:{size=31;break;}

}

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

printf("%d\t",i);day(sum+i);

}

}

return 0;

}

OUTPUT:

