/\* A. Array Manipulations using Pointers

One Dimensional Array: Write a C program to...

1. Find Mean, Median, Mode, Variance, Standard Deviation, and Range of 'n' elements in an

array \*/

#include<stdio.h>

int main()

{

int arr[50],\*p;

p=arr;

printf("Enter size : ");

int size;

scanf("%d",&size);

printf("Enter %d elements : ",size);

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

scanf("%d",p+i);

// code for finding mean...

float mean,sum=0.0;

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

sum+=\*(p+i);

mean=sum/size;

printf("Mean = %.2f",mean);

// code for sorting the array...

for(int round=1; round<size; round++)

{

for(int i=0; i<size-round; i++)

sum+=\*(p+i);

mean=sum/size;

printf("Mean = %.2f",mean);

// code for sorting the array...

for(int round=1; round<size; round++)

{

for(int i=0; i<size-round; i++)

{

if(\*(p+i)>\*(p+i+1))

{

int temp=\*(p+i);

\*(p+i)=\*(p+i+1);

\*(p+i+1)=temp;

}

}

}

// code to find the median...

float median;

if(size%2)

median=\*(p+size/2);

else

median=(\*(p+size/2)+\*(p+(size/2)-1))/2.0;

printf("\nMedian = %.2f",median);

// code to find the mode...

int C=0,mode;

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

{

int count=0;

for(int j=0; j<size; j++)

{

if(\*(p+i)==\*(p+j))

count++;

}

if(count>C)

{

C=count;

mode=\*(p+i);

}

}

if(C==1)

printf("\nNo Mode.");

else

printf("\nMode = %d",mode);

// code to find the variance...

sum=0.0;

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

sum+=(\*(p+i)-mean)\*(\*(p+i)-mean);

printf("\nVariance = %.2f",sum/size);

// code to find the standard deviation...

printf("\nStandard Deviation = %.2f",sqrt(sum/size));

// code to find the range...

printf("\nRange = %d",\*(p+size-1)-\*(p));

getch();

return 0;

}

Enter size : 3

Enter 3 elements : 34 67 89

Mean = 63.33

Median = 67.00

No Mode.

Variance = 510.89

Standard Deviation = 22.60

Range = 55

/\* A. Array Manipulations using Pointers

One Dimensional Array: Write a C program to...

2. Sort the 'n' elements of an array in Descending order \*/

#include<stdio.h>

int main()

{

int arr[50],\*p;

p=arr;

printf("Enter size : ");

int size;

scanf("%d",&size);

printf("Enter %d elements : ",size);

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

scanf("%d",p+i);

// code for sorting the array...

for(int round=1; round<size; round++)

{

for(int i=0; i<size-round; i++)

{

if(\*(p+i)<\*(p+i+1))

{

int temp=\*(p+i);

\*(p+i)=\*(p+i+1);

\*(p+i+1)=temp;

}

}

}

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

printf("%d ",\*(p+i));

getch();

return 0;

O/p:

Enter size : 3

Enter 3 elements : 45 98 99

99 98 45

/\* A. Array Manipulations using Pointers

One Dimensional Array: Write a C program to...

3. Find the second largest and smallest element in an array \*/

#include<stdio.h>

#include<limits.h>

int main()

{

int arr[50],\*p;

p=arr;

printf("Enter size : ");

int size;

scanf("%d",&size);

printf("Enter %d elements : ",size);

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

scanf("%d",p+i);

int l1,l2,s1,s2;

l1=l2=INT\_MIN;

s1=s2=INT\_MAX;

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

{

if(\*(p+i)>l1)

{

l2=l1;

l1=\*(p+i);

}

else if(\*(p+i)>l2 && \*(p+i)!=l1)

l2=\*(p+i);

if(\*(p+i)<s1)

{

s2=s1;

s1=\*(p+i);

}

else if(\*(p+i)<s2 &&\*(p+i)!=s1)

s2=\*(p+i);

}

printf("2ND Largest : %d",l2);

printf("\n2ND Smallest : %d",s2);

}

O/p:

Enter size : 5

Enter 5 elements : 67

89

99

78

45

2ND Largest : 89

2ND Smallest : 67

/\* A. Array Manipulations using Pointers

Two Dimensional Array: Write a C program to...

4. Print the leading diagonal, upper triangular and lower triangular elements of mxm array

\*/

int main()

{

int arr[5][5],(\*p)[5];

p=arr;

printf("Enter the size : ");

int size;

scanf("%d",&size);

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

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

{

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

{

for(int j=0; j<size; j++)

scanf("%d",\*(p+i)+j);

}

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

{

for(int j=0; j<size; j++)

{

printf("%d ",\*(\*(p+i)+j));

}

printf("\n");

}

printf("\nLeading Diagonal is : \n");

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

{

for(int j=0; j<size; j++)

{

if(i==j)

printf("%d",\*(\*(p+i)+j));

else

printf(" ");

}

printf("\n");

}

printf("\nUpper triangle is : \n");

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

{

for(int j=0; j<size; j++)

{

if(j>i)

printf("%d ",\*(\*(d",\*(\*(p+i)+j));

else

printf(" ");

}

printf("\n");

}

printf("\nUpper triangle is : \n");

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

{

for(int j=0; j<size; j++)

{

if(j>i)

printf("%d ",\*(\*(p+i)+j));

else

printf(" ");

printf("\n");

}

printf("\nLower triangle is : \n");

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

{

for(int j=0; j<size; j++)

{

if(j<i)

printf("%d ",\*(\*(p+i)+j));

else

printf(" ");

}

printf("\n");

}

}

O/p:

Enter the size: 2

Enter 2 \* 2 elements :

2

3

4

5

2 3

4 5

Leading Diagonal is :

2

5

Upper triangle is :

3

Lower triangle is :

4

/\* A. Array Manipulations using Pointers

Two Dimensional Array: Write a C program to...

5. Find the maximum & minimum element in each row and each coloumn of mxm array \*/

#include<stdio.h>

main()

{

int arr[5][5],(\*p)[5];

p=arr;

printf("Enter the size : ");

int size;

scanf("%d",&size);

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

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

for(int j=0; j<size; j++)

scanf("%d",\*(p+i)+j);

}

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

{

for(int j=0; j<size; j++)

{

printf("%d ",\*(\*(p+i)+j));

}

printf("\n");

}

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

{

int max=0;

for(int j=0; j<size; j++)

{

if(max< \*(\*(p+i)+j) )

max= \*(\*(p+i)+j);

}

printf("Max in row %d : %d\n",i,max);

}

printf("\n");

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

{

int max=0;

for(int j=0; j<size; j++)

{

if(max< \*(\*(p+j)+i) )

max= \*(\*(p+j)+i);

}

printf("Max in column %d : %d\n",i,max);

}

printf("\n");

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

{

int min=\*(\*(p+i));

for(int j=0; j<size; j++)

{

if(min> \*(\*(p+i)+j) )

min= \*(\*(p+i)+j);

}

printf("MIN in row %d : %d\n",i,min);

}

printf("\n");

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

{

int min=\*(\*(p)+i);

for(int j=0; j<size; j++)

{

if(min> \*(\*(p+j)+i) )

min= \*(\*(p+j)+i);

}

printf("MIN in column %d : %d\n",i,min);

getch();

return 0;

}

Output:

Enter the size : 3

Enter 3 \* 3 elements : 4

5

6

7

8

9

8

6

2 4 5

6 7 8

9 8 6

Max in row 0 : 5

Max in row 1 : 8

Max in row 2 : 9

Max in column 0 : 9

Max in column 1 : 8

Max in column 2 : 8

MIN in row 0 : 2

MIN in row 1 : 6

MIN in row 2 : 6

MIN in column 0 : 2

MIN in column 1 : 4

MIN in column 2 : 5

/\* A. Array Manipulations using Pointers

Two Dimensional Array: Write a C program to...

6. Perform matrix multiplication between two mxm array \*/

#include<stdio.h>

int main()

{

int arr[5][5],arr1[5][5],(\*p)[5],(\*p1)[5];

p=arr;

p1=arr1;

printf("Enter the size : ");

int size;

scanf("%d",&size);

printf("Enter %d \* %d elements for matrix 1 : ",size,size);

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

{

for(int j=0; j<size; j++)scanf("%d",\*(p+i)+j);

}

printf("Enter %d \* %d elements for matrix 2 : ",size,size);

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

{

for(int j=0; j<size; j++)

scanf("%d",\*(p1+i)+j);

}

printf("1st matrix is : \n");

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

{

for(int j=0; j<size; j++)

{

printf("%d ",\*(\*(p+i)+j));

}

printf("\n");

}

printf("2nd matrix is : \n");

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

{

for(int j=0; j<size; j++)

{

printf("%d ",\*(\*(p1+i)+j));

}

printf("\n");

}

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

{

for(int j=0; j<size; j++)

{

int sum=0;

for(int k=0; k<size; k++)

{

sum+=(\*(\*(p+i)+k)) \* (\*(\*(p1+k)+j));

}

printf("%d ",sum);

}

printf("\n");

}

getch();

return 0;

}

Output:

Enter the size : 2

Enter 2 \* 2 elements for matrix 1 :

5

9

8

7

Enter 2 \* 2 elements for matrix 2 :

7

9

0

7

1st matrix is :

5 9

8 7

2nd matrix is :

7 9

0 7

35 108

56 121

/\* B. String Manipulations using Pointers

7. Write a C Program to convert

a. Upper case to Lower case,

b. Lower case to Upper case,

c. Togglecase,

d. Sentance case

\*/

#include<stdio.h>

int main()

{

char \*a,str[20];

printf("Enter a string : ");

a=str;

gets(a);

puts(strupr(a));

puts(strlwr(a));

for(int i=0; \*(a+i)!='\0'; i++)

{

if(i==0)

continue;

while(\*(a+i)==32)

i+=2;

if(\*(a+i)>=97 && \*(a+i)<=122)

{

\*(a+i)-=32;

}

}

puts(a);

for(int i=0; \*(a+i)!='\0'; i++)

{

while(\*(a+i)==32)

i++;

if(i==0)

if(\*(a+i)>=97 && \*(a+i)<=122)

\*(a+i)-=32;

if(\*(a+i)=='.')

{

printf("hello\n");

i++;

while(\*(a+i)==32)

i++;

printf(" %d ",i);

if(\*(a+i)>=97 && \*(a+i)<=122)

{

\*(a+i)=\*(a+i)-32;

i++;

}

}

if(i!=0 && \*(a+i)>=65 && \*(a+i)<=90)

\*(a+i)+=32;

}

Puts(a);

}

/\*B. String Manipulations using Pointers

8. Write a C Program to read 2 string constants into a and b.

Compare whether they are equal or not. if not, join them together.

Then copy the contents of a to the variable c.

At the end of the program, print the contents of all three variables and their length. (With

and Without String Handling Functions).

\*/

#include<stdio.h>

int main()

{

char str[20],str1[20],str3[20],\*a,\*b,\*c;

a=str;

b=str1;

c=str3;

printf("Enter a string : ");

gets(a);

printf("Enter another string : ");

gets(b);

printf("Using string handling functions...\n");

if(strcmp(a,b))

strcpy(c,a);

strcat(a,b);

}

printf("a = %s and size = %d\nb = %s and size = %d\nc = %s and size = %d\n",a,strlen(a),b,strlen(b),c,strlen(c));

printf("\nWithout using string handling functions...");

a=str;

b=str1;

c=str3;

printf("\nEnter a string : ");

gets(a);

printf("Enter another string : ");

gets(b);

int i;

for(i=0; \*(a+i)!='\0'; i++)

{

if(\*(a+i) != \*(b+i))

{

i=-1;

break;

}

}

if(i=-1)

{

int l1=strlen(a);

int l2=strlen(b);

for(i=0; \*(a+i)!='\0'; i++)

\*(c+i)=\*(a+i);

\*(a+l1)=32;

int j=0;

for(int i=l1+1; i<=l1+l2; i++,j++)

\*(a+i)=\*(b+j);

}

printf("\na = %s and size = %d\nb = %s and size = %d\nc = %s and size = %d\n",a,strlen(a),b,strlen(b),c,strlen(c));

}

O/P:

Enter a string : its devil

Enter another string : data structure

Using string handling functions...

a = its devil data structure and size = 27

b = data structure and size = 14

c = its devil and size = 13

Without using string handling functions...

Enter a string : welcome

Enter another string : programming

a = welcome programming structure and size = 27

b = programming and size = 11

c = welcomedev vil and size = 13

/\* B. String Manipulations using Pointers

9. Write a C program to read a string and prints if it is a palindrome or not. \*/

#include<stdio.h>

#include<string.h>

int main()

{

char str[20],copy[20],temp,\*p;

int i,l;

printf("Enter a string : ");

p=str;

gets(p);

strcpy(copy,str);

for(l=0; \*(p+l)!='\0'; l++);

for(i=0; i<l/2; i++)

{

temp=\*(p+i);

\*(p+i)=(\*(p+l-i-1));

\*(p+l-i-1)=temp;

}

if(strcmp(p,copy))

printf("Not pallindrome");

else

printf("Pallindrome");

getch();

return 0;

}

Output:

Enter a string : its devil

Not palindrome

/\* C. Functions using Pointers Write a C Program (Using Call by Value, Call by Reference &

Category of Functions)

10. Check Prime and Armstrong Number by making function \*/

int main()

{

int n;

printf("Enter a number : ");

scanf("%d",&n);

primeCBV(n);

printf("\n");

armstrong(n);

getch();

return 0;}

void primeCBV(int n)

{

int i;

for(i=2; i<n; i++)

{

if(n%i==0)

break;

}

if(i==n)

printf("Prime");

else

printf("Not Prime");

}

void primeCBR(int \*n)

{

int i;

for(i=2; i<\*n; i++)

{

if(\*n%i==0)

break;

}

if(i==\*n)

printf("Prime");

else

printf("Not Prime");

}

void armstrong(int n)

{

int num=0,t=n;

while(n)

{

int r=n%10;

num=num+(r\*r\*r);

n=n/10;

}

if(t==num)

printf("\nArmstrong");

else

printf("\nNot Armstrong");

}

Output:

Enter a number : 234

Not Prime

Not Armstrong

/\*C. Functions using Pointers

Write a C Program (Using Call by Value, Call by Reference & Category of Functions)

11. Reverse a sentence using String Functions \*/

#include<stdio.h>

#include<string.h>

char \*revSen(char \*s);

int main()

{

printf("Enter a sentence : ");

char str[50];

gets(str);

revSentence(str);

puts(str);

char \*s=revSen(str);

puts(s);

}

void revSentence(char str[])

{

strrev(str);

}

char \*revSen(char \*s)

{

strrev(s);

return s;

}

/\*C. Functions using Pointers

Write a C Program (Using Call by Value, Call by Reference & Category of Functions)

12. Calculate the power of a number using recursion\*/

#include<stdio.h>

int power(int,int);

int (\*func\_pointer)(int,int);

int main()

{

printf("Enter a number : ");

int n;

scanf("%d",&n);

printf("Enter power of %d : ",n);

int p;

scanf("%d",&p);

func\_pointer=&power;

printf("%d",(\*func\_pointer)(n,p));

}

int power(int n,int p)

{

if(p==0)

return 1;

return(n\*power(n,p-1));

}

Enter a number : 45

Enter power of 45 : 2

2025

/\* D. Structures using Pointers Write a C Program,

13. Store Information(name, roll and marks) of a Student Using Structure \*/

struct student

{

char name[20];

int rollno;

float marks;

};

int main()

{

struct student s1,\*sp;

sp=&s1;

printf("Enter name of the student : ");

gets(sp->name);

printf("Enter roll no : ");

scanf("%d",&sp->rollno);

printf("Enter mark : ");

scanf("%f",&sp->marks);

printf("Name : %s\nRoll No : %d\nMark : %f",sp->name,sp->rollno,sp->marks);

getch();

return 0;

}

Output:

Enter name of the student : Pinaki

Enter roll no : 1000

Enter mark : 600

Name : Gopal

Roll No : 1000

Mark : 600.000000

/\* D. Structures using Pointers Write a C Program,

14. Add Two Complex Numbers by Passing Structure to a Function \*/

#include<stdio.h>

struct complex

{

int real;

int imag;

};

struct complex \* add(struct complex \*p,struct complex \*p1);

int main()

{

struct complex a,b,\*p,\*p1;

p=&a; p1=&b;

printf("Enter first complex number : ");

printf("Enter real part : ");

scanf("%d",&p->real);

printf("Enter imaginary part : ");

scanf("%d",&p->imag);

printf("Enter second complex number : ");

printf("Enter real part : ");

scanf("%d",&p1->real);

printf("Enter imaginary part : ");

scanf("%d",&p1->imag);

p=add(p,p1);

printf("Addition : %d + %d i",p->real,p->imag);

getch();

return 0;

}

struct complex \* add(struct complex \*p,struct complex \*p1)

{

struct complex \*n,b;

n=&b;

n->real=p->real+p1->real;

n->imag=p->imag+p1->imag;

return n;

}

Output:

Enter first complex number : Enter real part : 34

Enter imaginary part : 78

Enter second complex number : Enter real part : 99

Enter imaginary part : 77

Addition : 133 + 155 i

/\* D. Structures using Pointers Write a C Program,

15. Store & Retrieve Information, Calculate Total, Average and Rank of 10 Students Using

Structure

\*/

#include<stdio.h>

typedef struct

char name[20];

int m1;

int m2;

int m3;

}students;

int main()

{

students s[10],\*p;

p=s;

printf("Enter details of 10 students : \n");

for(int i=0; i<10; i++)

{

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

printf("Name : ");

fflush(stdin);

gets(p[i].name);

printf("Enter mark 1 : ");

scanf("%d",&(p+i)->m1);

printf("Enter mark 2 : ");

scanf("%d",&(p+i)->m2);

printf("Enter mark 3 : ");

scanf("%d",&(p+i)->m3);

for(int i=0; i<10; i++)

{

int sum=0;

sum=(p+i)->m1+(p+i)->m2+(p+i)->m3;

printf("%s\nSum of marks : %d, Average : %f",(p+i)->name,sum,sum/3.0);

float per=(sum\*100)/600;

if(per>=60)

printf(" Rank 1");

else if(per>=50 && per<60)

printf(" Rank 2");

else if(per>=30)

printf(" Rank 3");

else

printf(" Fail");

printf("\n");

}

}

Output:

Enter details of 10 students :

Enter details of student 1 :

Name :pinaki

Enter mark 1 : 67

Enter mark 2 : 54

Enter mark 3 : 44

Enter details of student 2 :

Name :Dolo

Enter mark 1 :52

Enter mark 2 : 41

Enter mark 3 : 53

Enter details of student 3 :

Name : Rahul

Enter mark 1 : 23

Enter mark 2 : 45

Enter mark 3 : 67

Enter details of student 4 :

Name :Bonty

Enter mark 1 : 42

Enter mark 2 : 23

Enter mark 3 : 56

Enter details of student 5 :

Name : ankita

Enter mark 1 : 33

Enter mark 2 : 22

Enter mark 3 : 89

Enter details of student 6 :

Name : ankita

Enter mark 1 : 33

Enter mark 2 : 22

Enter mark 3 : 89