Crash Course on "Programming in C with Data Structure perspective"

Homework Solutions

**Session 1 – C Basics , Conditional statements & Looping statements – 21.11.2022**

| Q1 | Develop a simple calculator using conditional statements of C |
|---|---|

```c
#include <stdio.h>
#include <math.h>

int main()
{
  int a,b,ans,flag=1;
  char op;

  do
  {
    printf("Enter first Operator : ");
    scanf("%d",&a);
    printf("Enter second Operator : ");
    scanf("%d",&b);
    printf("Choose  a  Operator (+,-,*,/,^) : ");
    scanf(" %c",&op);
    switch(op)
    {
      case '+':
      {
        ans=a+b;
        printf("Sum = %d\n\n",ans);
        break;
      }
      case '-':
      {
        ans=a-b;
        printf("Difference = %d\n\n",ans);
        break;
      }
      case '*':
      {
        ans=a*b;
        printf("Product = %d\n\n",ans);
        break;
      }
      case '/':
```

```
      {
        float sol=(float)a/(float)b;
        printf("Divison = %.2f\n\n",sol);
        break;
      }
      case '%':
      {
        ans=a%b;
        printf("Reminder %d= \n\n",ans);
        break;
      }
      case '^':
      {
        ans=pow(a,b);
        printf("%d to the power of %d = %d\n\n",a,b,ans);
        break;
      }
    }
    printf("If you want continue press one : ");
    scanf("%d",&flag);
  }while(flag==1);

  return 0;
}
```

## Sample Input & Output:

```
Enter first Operator : 10
Enter second Operator : 5
Choose  a  Operator (+,-,*,/,^) : +
Sum = 15

If you want continue press one : 1
Enter first Operator : 10
Enter second Operator : 3
Choose  a  Operator (+,-,*,/,^) : /
Divison = 3.33

If you want continue press one : 1
Enter first Operator : 2
Enter second Operator : 4
Choose  a  Operator (+,-,*,/,^) : ^
2 to the power of 4 = 16

If you want continue press one : 0
```

| Q2 | Using loops in C implement the following<br>. Find factorial of a number<br>. Find sum of first ‘n’ numbers |
|---|---|

a)

```c
#include <stdio.h>

int main()
{
    int n, i, fact=1;
    printf("Enter positive integer n : ");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
    {
        fact=fact*i;
    }
    printf("Factorial is %d", fact);
    return 0;

}
```

b)

```c
#include <stdio.h>

int main()
{
    int n, i, sum=0;
    printf("Enter positive integer n : ");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
    {
        sum=sum+i;
    }
    printf("Total sum is %d", sum);
    return 0;

}
```

**Sample Input & Output:**

```
Enter positive integer n : 5
Factorial is 120
```

```
Enter positive integer n : 5
Total sum is 15
```

## Session 2 – Loops and Nested Loops – 22.11.2022

| Q1 | Print only odd numbers in a range of ‘n’. Use continue statement. |
|---|---|

```c
#include<stdio.h>

void main(){
  int n,i;
  printf("Enter a number: ");
  scanf("%d",&n);
  printf("the odd numbers in a given range:\n");
  for(i=0;i<=n;i++){
    if(i%2==0)
      continue;
    else
      printf("%d  ",i);
  }
  printf("\n");
}
```

### Sample Input & Output:

```
Enter a number: 10
the odd numbers in a given range:
1  3  5  7  9
```

| Q2 | Print the following pattern<br><br>5<br>5 4<br>5 4 3<br>5 4 3 2<br>5 4 3 2 1 |
|---|---|

```c
#include <stdio.h>
void main()
{
  int n=5;
  for(int i=0;i<n;i++)
  {
      for(int j=0;j<=i;j++)
        printf("%d ",n-j);
      printf("\n");
  }
}
```

| Q3 | Print the prime numbers in a given range.<br>Hint: Prime numbers are numbers that are divisible only by 1 and itself. |
|---|---|

```c
#include<stdio.h>

void main(){
  int n,i,j,count;
  printf("Enter a number: ");
  scanf("%d",&n);
  printf("the prime numbers in a given range:\n");
  for(i=2;i<=n;i++){
    count=0;
    for(j=1;j<=i;j++){
      if(i%j==0){
        count++;
       }
       else
        continue;
     }
     if(count==2){
        printf("%d ",i);
     }
  }
  printf("\n");
}
```

**Sample Input & Output:**

```
Enter a number: 10
the prime numbers in a given range:
2 3 5 7
```

| Q4 | Write a program to display multiplication table from 1 to 10. |
|---|---|

```c
#include <stdio.h>
void main()
{
  int i, n;
  printf("\nEnter the mult table you want: ");
  scanf("%d", &n);
  for(i=1;i<=10;i++)
    printf("%d*%d=%d\n", i, n, i*n);
}
```

**Sample Input & Output:**

```
Enter the mult table you want: 2
1*2=2
2*2=4
3*2=6
4*2=8
5*2=10
6*2=12
7*2=14
8*2=16
9*2=18
10*2=20
```

## Session 3 – Arrays – 23.11.2022

| Q1 | Perform matrix multiplication in C. |
|---|---|

```c
#include<stdio.h>
#include<stdlib.h>
int main(){

    int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;

    printf("enter the number of row=");
    scanf("%d",&r);

    printf("enter the number of column=");
    scanf("%d",&c);

    printf("enter the first matrix element=\n");
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            scanf("%d",&a[i][j]);

    printf("enter the second matrix element=\n");
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
            scanf("%d",&b[i][j]);

    printf("multiply of the matrix=\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            mul[i][j]=0;
            for(k=0;k<c;k++)
            {
                mul[i][j]+=a[i][k]*b[k][j];
            }
```

```
        }
    }

    //for printing result
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("%d\t",mul[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

**Sample Input & Output:**

```
enter the number of row=3
enter the number of column=3
enter the first matrix element=
1 1 1
2 2 2
3 3 3
enter the second matrix element=
1 1 1
2 2 2
3 3 3
multiply of the matrix=
6       6       6
12      12      12
18      18      18
```

| Q2 | Given a matrix print the transpose of the matrix. |
|---|---|

```
#include <stdio.h>

void main()
{
  int a[10][10],b[10][10],row,col,i,j;
  printf("Enter a no.of row= ");
  scanf("%d",&row);
  printf("Enter a no.of col = ");
  scanf("%d",&col);

  //Read Matrix A
  for(i=0;i<row;i++)
  {
```

```
    for(j=0;j<col;j++)
    {
      printf("Enter a element a[%d][%d]=",i,j);
      scanf("%d",&a[i][j]);
    }
    printf("\n");
  }
  printf("\n");

  //Assign A^-1 in B
  for(i=0;i<row;i++)
  {
    for(j=0;j<col;j++)
      b[j][i]=a[i][j];
  }


  //Print Matrix A
  for(i=0;i<row;i++)
  {
    for(j=0;j<col;j++)
      printf("%d ",a[i][j]);
    printf("\n");
  }
  printf("\n");

  //print Inverse MAtrix A
  for(i=0;i<col;i++)
  {
    for(j=0;j<row;j++)
       printf("%d ",b[i][j]);
    printf("\n");
  }
}
```

**Sample Input & Output:**

```
Enter a no.of row= 3
Enter a no.of col = 2
Enter a element a[0][0]=1
Enter a element a[0][1]=2

Enter a element a[1][0]=3
Enter a element a[1][1]=4

Enter a element a[2][0]=5
Enter a element a[2][1]=6



1 2
3 4
5 6

1 3 5
2 4 6
```

| Q3 | Insert an element in the beginning, middle, and end of an array. |
|---|---|

```c
#include <stdio.h>
void printArray(int a[],int n)
{
  for(int i=0;i<n;i++)
    printf("%d ",a[i]);
}

void main()
{
  int i,j,k,n,a[10],op;
  printf("\nEnter No of element in the array = ");
  scanf("%d",&n);

  //Read Array:
  for(i=0;i<n;i++)
  {
    printf("Enter a element a[%d] = ",i);
    scanf("%d",&a[i]);
  }
  //Print Array:
  printArray(a,n);

  printf("\n\n***** WELCOME TO ARRAY OPERATIONS *****");
  printf("\n\n 1. Insert at Begining\n 2. Insert at Middle\n 3. Insert at
End\n 4. Exit\n");

  do{
    printf("\n\nChoose one of the option = ");
    scanf("%d",&op);

    switch(op)
    {
      case 1:
      {
        int data;
        printf("Enter insert element at Begining : ");
        scanf("%d",&data);
        for(i=n;i>0;i--)
          a[i]=a[i-1];
        a[0]=data;
        n++;
        printArray(a,n);
        break;
      }

      case 2:
      {
        int data,pos;
        printf("Enter insert element at middle : ");
```

```
        scanf("%d",&data);
        printf("Enter insert position : ");
        scanf("%d",&pos);
        for(i=n;i>pos;i--)
          a[i]=a[i-1];
        a[pos]=data;
        n++;
        printArray(a,n);
        break;
      }

      case 3:
      {
        int data;
        printf("Enter insert element at End : ");
        scanf("%d",&data);
        a[n]=data;
        n++;
        printArray(a,n);
        break;
      }
    }
  }while(op!=4);
  printf("\nThank you !...\n");
}
```

**Sample Input & Output:**

```
Enter No of element in the array = 2
Enter a element a[0] = 1
Enter a element a[1] = 3

Array:1 3

***** WELCOME TO ARRAY OPERATIONS *****

 1. Insert at Begining
 2. Insert at Middle
 3. Insert at End
 4. Exit


Choose one of the option = 1
Enter insert element at Begining : 0
0 1 3

Choose one of the option = 2
Enter insert element at middle : 2
Enter insert position : 2
0 1 2 3

Choose one of the option = 3
Enter insert element at End : 4
0 1 2 3 4

Choose one of the option = 4

Thank you !...
```

| Q4 | Delete an element from the array. |
|---|---|

```c
#include <stdio.h>
void printArray(int a[],int n)
{
  for(int i=0;i<n;i++)
    printf("%d ",a[i]);
  printf("\n");
}

void main()
{
  int i,j,k,n,a[10],ele;
  printf("\nEnter No of element in the array = ");
  scanf("%d",&n);

  //Read Array:
  for(i=0;i<n;i++)
  {
    printf("Enter a element a[%d] = ",i);
    scanf("%d",&a[i]);
  }
    //Print Array:
  printArray(a,n);

    printf("Enter Element to delete: ");
    scanf("%d",&ele);

    for(j=0;j<n;j++)
        if(a[j]==ele)
            k=j;
    for(i=k;i<n;i++)
        a[i]=a[i+1];
    a[n-1]=0;
    n--;
    printArray(a,n);
}
```

**Sample Input & Output:**

```
Enter No of element in the array = 4
Enter a element a[0] = 1
Enter a element a[1] = 2
Enter a element a[2] = 3
Enter a element a[3] = 4
1 2 3 4
Enter Element to delete: 2
1 3 4
```

## Session 4 – Array operations & Sorting Techniques – 24.11.2022

| | |
|---|---|
| Q1 | Write a menu-driven C program to implement all the array operations discussed in last session. In addition, also include selection sort and bubble sort of array. |

```c
#include <stdio.h>
void printArray(int a[],int n)
{
  for(int i=0;i<n;i++)
    printf("%d ",a[i]);
}

void main()
{
  int i,j,k,n,a[10],op;
  printf("\nEnter a No of element in the array = ");
  scanf("%d",&n);

  //Read Array:
  for(i=0;i<n;i++)
  {
    printf("Enter a element a[%d] = ",i);
    scanf("%d",&a[i]);
  }
  //Print Array:
  printf("\nArray:");
  for(i=0;i<n;i++)
    printf("%d ",a[i]);

  printf("\n\n***** WELCOME TO ARRAY OPERATIONS *****");
  printf("\n\n 1. Insert at Begining\n 2. Insert at Middle\n 3. Insert at
End");
  printf("\n 4. Delete\n 5. Selection Sort\n 6. Bubble Sort\n 7. Exit\n");

  do{
    printf("\n\nChoose one of the option = ");
    scanf("%d",&op);

    switch(op)
    {
      case 1:
      {
        int data;
        printf("Enter a insert element at Begining : ");
        scanf("%d",&data);
        for(i=n;i>0;i--)
          a[i]=a[i-1];
        a[0]=data;
        n++;
        printArray(a,n);
```

```
      break;
    }

    case 2:
    {
      int data,pos;
      printf("Enter a insert element at middle : ");
      scanf("%d",&data);
      printf("Enter a insert position : ");
      scanf("%d",&pos);
      for(i=n;i>pos;i--)
        a[i]=a[i-1];
      a[pos]=data;
      n++;
      printArray(a,n);
      break;
    }

    case 3:
    {
      int data;
      printf("Enter a insert element at End : ");
      scanf("%d",&data);
      a[n]=data;
      n++;
      printArray(a,n);
      break;
    }

    case 4:
    {
      int ele;
      printf("Enter Element to delete: ");
      scanf("%d",&ele);

      for(j=0;j<n;j++)
          if(a[j]==ele)
              k=j;
      for(i=k;i<n;i++)
          a[i]=a[i+1];
      a[n-1]=0;
      n--;
      printArray(a,n);
      break;
    }

    case 5:
    {
      for(i=0;i<n-1;i++)
      {
          int index=i;
          for(j=i+1;j<n;j++)
```

```
            {
              if(a[index]>a[j])
                index=j;
            }
            if(index!=i)
            {
              int temp=a[i];
              a[i]=a[index];
              a[index]=temp;
            }
        }
        printArray(a,n);
        break;
      }

      case 6:
      {
          int temp;
        for(i=0;i<n;i++)
      {
        for(j=i+1;j<n;j++)
        {
          if(a[i]>a[j])
          {
            temp=a[j];
            a[j]=a[i];
            a[i]=temp;
          }
        }
      }
      printArray(a,n);
        break;
      }


    }
  }while(op!=7);
}
```

**Sample Input & Output:**

```
Enter a No of element in the array = 5
Enter a element a[0] = 6
Enter a element a[1] = 8
Enter a element a[2] = 3
Enter a element a[3] = 1
Enter a element a[4] = 6

Array:6 8 3 1 6

***** WELCOME TO ARRAY OPERATIONS *****

 1. Insert at Begining
 2. Insert at Middle
 3. Insert at End
 4. Delete
 5. Selection Sort
 6. Bubble Sort
 7. Exit


Choose one of the option = 6
1 3 6 6 8

Choose one of the option = 7
```

## Session 5 – Strings – 25.11.2022

| | |
|---|---|
| Q1 | Find the length of strings without using the built-in function. |

```c
#include<stdio.h>

void main(){
   char str[50];
   int i;

   printf("Enter a string1: ");
   scanf("%s",str);

   i=0;
   while(str[i]!='\0'){
     i++;
   }
   printf("The length of the string is %d\n",i);
}
```

**Sample Input & Output:**

```
Enter a string1: hello
The length of the string is 5
```

| Q2 | Implement string concatenation of two strings without using a built-in function. |
|---|---|

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>

void main(){
   char str[50],str1[50],str2[50];
   int i,j;

   printf("Enter a string1: ");
   scanf("%s",str);
   printf("Enter a string2: ");
   scanf("%s",str1);

   i=0;
   // erases the data in the n bytes of the memory if already present
   bzero(str2,sizeof(str2));

   while(str[i]!='\0'){
     str2[i]=str[i];
     i++;
   }
   str2[i++]=' ';
   j=0;
   while(str1[j]!='\0'){
     str2[i]=str1[j];
     i++;
     j++;
   }

   printf("The string 1 is %s\n",str);
   printf("The string 2 is %s\n",str1);
   printf("The string 3 is %s\n",str2);
}
```

**Sample Input & Output:**

```
Enter a string1: hello
Enter a string2: worlds
The string 1 is hello
The string 2 is worlds
The string 3 is hello worlds
```

| Q3 | Implement string reversal without using a built-in function. |
|---|---|

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>

void main(){
   char str[50],str1[50];

   int i,j,n,count=0;

   printf("Enter a string: ");
   scanf("%s",str);
   printf("The string is %s\n",str);

   i=0;
   bzero(str1,sizeof(str1));

   while(str[i]!='\0'){
     i++;
   }

  for(j=0;j<i;j++){
     str1[j]=str[i-j-1];
   }
   printf("The Reversed string is %s\n",str1);
}
```

**Sample Input & Output:**

```
Enter a string: programming
The string is programming
The Reversed string is gnimmargorp
```

| Q4 | Check whether the given string is palindrome or not. |
|---|---|

```c
#include<stdio.h>


void main(){
   char str[50],str1[50];

   int i,n,count=0;

   printf("Enter a string: ");
   scanf("%s",str);
   printf("The string is %s\n",str);
```

```
   n=0;
   while(str[n]!='\0'){
     n++;
   }
  for(i=0;i<n;i++){
     if(str[i]==str[n-i-1]){
          count++;
   }
  }
  if(count==n)
    printf("The string %s is palindrome\n",str);
  else
    printf("The string %s is not palindrome\n",str);

}
```

**Sample Input & Output:**

```
Enter a string: malayalam
The string is malayalam
The string malayalam is palindrome
```

**Session 6 – Functions – 26.11.2022**

| | |
|---|---|
| Q1 | Write C functions to implement the calculator module. Prompt user for the operation. |

```
#include<stdio.h>

int add(int a, int b){
  return (a+b);
}

int sub(int a, int b){
  return (a-b);
}

int mul(int a, int b){
  return (a*b);
}

int div(int a, int b){
  return (a/b);
}

int main()
{
```

```
  char o;
  int a, b;
  printf ("Enter operation: ");
  scanf (" %c", &o);
  printf ("Enter a and b (hit ',' key between entering 'a' and 'b')\n");
  scanf ("%d, %d", &a, &b);

  switch(o)
  {
    case '+':
    {
        int sum = add(a, b);
        printf("\nSum = %d", sum);
        break;
    }

    case '-':
    {
        int difference = sub(a, b);
        printf("\ndif = %d", difference);
        break;
    }

    case '*':
    {
        int product = mul(a, b);
        printf("\npro = %d", product);
        break;
    }

    case '/':
    {
        int quotient = div(a, b);
        printf("\ndiv = %d", quotient);
        break;
    }
  }
}
```

## Sample Input & Output:

```
Enter operation: *
Enter a and b (hit ',' key between entering 'a' and 'b')
5,3

pro = 15
```

| Q2 | Write a function to implement selection sort of an array.<br>Note: Array must be a parameter to the function. Array size should be user input. |
|---|---|

```c
#include <stdio.h>
void selSort(int a[],int n)
{
  for(int i=0;i<n-1;i++)
  {
    int index=i;
    for(int j=i+1;j<n;j++)
    {
      if(a[index]>a[j])
        index=j;
    }
    if(index!=i)
    {
      int temp=a[i];
      a[i]=a[index];
      a[index]=temp;
    }
  }
}

void main()
{
  int n,i,j,a[10];
  printf("Enter a no.of Elements : ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    printf("Enter a element a[%d] : ",i);
    scanf("%d",&a[i]);
  }

  printf("\nBefore Sorting:\n");
  for(int i=0;i<n;i++)
    printf("%d ",a[i]);
  printf("\n\n");

  selSort(a,n);

  printf("\nAfter Sorting:\n");
  for(int i=0;i<n;i++)
    printf("%d ",a[i]);
  printf("\n\n");
}
```

**Sample Input & Output:**

```
Enter a no.of Elements : 5
Enter a element a[0] : 6
Enter a element a[1] : 9
Enter a element a[2] : 2
Enter a element a[3] : 3
Enter a element a[4] : 1

Before Sorting:
6 9 2 3 1


After Sorting:
1 2 3 6 9
```

## Session 7 – Recursion – 28.11.2022

| | |
|---|---|
| Q1 | Write a recursive C function to find the sum of digits of a number. |

```c
#include <stdio.h>

int sumOfDig(int n)
{

  int sum;

  if(n==0){
      return 0;
  }

  sum=(n%10);
  return (sum+sumOfDig((n/10)));

}

void main()
{
  int n;
  printf("Enter n: ");
  scanf("%d", &n);
  printf("\nSum of digits of input number '%d' = %d", n, sumOfDig(n));
}
```

**Sample Input & Output:**

```
Enter n: 153

Sum of digits of input number '153' = 9
```

| Q2 | Implement binary search technique using simple loops. |
|---|---|

```c
#include<stdio.h>

int binarysearch(int a[],int start,int end,int x){
  int middle;

  while(start<=end){
    middle=start+(end-start)/2;
    if(a[middle]==x)
      return middle;
    else if(x>a[middle])
      start=middle+1;
    else
      end=middle-1;
  }
  return -1;
}
void main(){
  int a[20],n,first,last,index,i,ele;

  printf("Enter no of Elements in array: ");
  scanf("%d",&n);

  for(i=0;i<n;i++){
    printf("Element: ");
    scanf("%d",&a[i]);
  }
  printf("enter a element to be searched: ");
  scanf("%d",&ele);
  index=binarysearch(a,0,n-1,ele);
  if(index==-1)
    printf("The element %d is not found in array\n",ele);
  else
    printf("The element %d is found in position %d\n",ele,index);
}
```

**Sample Input & Output:**

```
Enter no of Elements in array: 5
Element: 6
Element: 9
Element: 3
Element: 2
Element: 5
enter a element to be searched: 3
The element 3 is found in position 2
```

| Q3 | Write a recursive C function to implement binary search in an array. |
|---|---|

```c
#include<stdio.h>

int binarysearch(int a[],int start,int end,int x){
  int middle;
  if(start<=end)
  {
    middle=start+(end-start)/2;
    if(a[middle]==x)
      return middle;
    else if(x>a[middle])
      return binarysearch(a,start+1,end,x);
    else
      return binarysearch(a,start,end-1,x);
  }
  else
      return -1;
}
void main(){
  int a[20],n,first,last,index,i,ele;

  printf("Enter no of Elements in array: ");
  scanf("%d",&n);

  for(i=0;i<n;i++){
    printf("Element: ");
    scanf("%d",&a[i]);
  }
  printf("enter a element to be searched: ");
  scanf("%d",&ele);
  index=binarysearch(a,0,n-1,ele);
  if(index==-1)
    printf("The element %d is not found in array\n",ele);
  else
    printf("The element %d is  found in %d\n",ele,index);
}
```

**Sample Input & Output:**

```
Enter no of Elements in array: 5
Element: 2
Element: 4
Element: 6
Element: 8
Element: 9
enter a element to be searched: 8
The element 8 is  found in 3
```

| Q4 | Write a recursive C function to find the binary number of a given decimal number. |
|---|---|

```c
#include<stdio.h>

int binary(int n){
   if(n==0)
     return 0;
   else
     return (n%2)+ 10*binary(n/2);
}


int main(){
  int n,b;
  printf("Enter decimal no: ");
  scanf("%d",&n);

  b=binary(n);
  printf("The binary of %d is %d\n",n,b);
}
```

**Sample Input & Output:**

```
Enter decimal no: 11
The binary of 11 is 1011
```

| Q5 | Write a recursive C function to find the LCM and GCD of 2 numbers. |
|---|---|

```c
#include<stdio.h>

int gcd(int n1,int n2){
   if(n1==n2)
     return n1;
   else if(n1>n2)
     return gcd(n2,n1-n2);
   else
      return gcd(n1,n2-n1);
}

int lcm(int n1,int n2){
   static int max=1;
   if(max%n1==0 && max%n2==0)
     return max;
   else{
     max++;
     return lcm(n1,n2);
   }
```

```c
}


int main(){
  int n1,n2,gcdn,lcmn;
  printf("Enter n1: ");
  scanf("%d",&n1);
  printf("Enter n1: ");
  scanf("%d",&n2);
  gcdn=gcd(n1,n2);
  printf("The gcd of %d and %d is %d\n",n1,n2,gcdn);
  lcmn=lcm(n1,n2);
  printf("The lcm of %d and %d is %d\n",n1,n2,lcmn);
}
```

**Sample Input & Output:**

```
Enter n1: 6
Enter n1: 3
The gcd of 6 and 3 is 3
The lcm of 6 and 3 is 6
```

## Session 8 – Pointers – 29.11.2022

| | |
|---|---|
| Q1 | Write a program in C to print all permutations of a given string using pointers. |

```c
#include <stdio.h>
#include <string.h>
void swap(char *a,char *b)
{
  char *temp=*a;
  *a=*b;
  *b=temp;
}

void permute(char *a,int l,int r)
{
  if(l==r)
    printf("%s ",a);
  else
  {
    for(int i=l;i<=r;i++)
    {
      swap((a+l),(a+i));
      permute(a,l+1,r);
      swap((a+l),(a+i));
    }
  }
```

```
}

void main()
{
  char *str;
  char input[10]="ABC";
  str=&input;
  int n=strlen(input);
  permute(str,0,n-1);
  printf("\n\n");
}
```

**Sample Input & Output:**

```
ABC ACB BAC BCA CBA CAB
```

| | |
|---|---|
| Q2 | Write a program in C to find the largest element using Dynamic Memory Allocation. |

```
#include <stdio.h>
#include <stdlib.h>

int findLarge(int *element,int n)
{
  int large=*element;
  for(int i=1;i<n;i++)
  {
    if(large<*(element+i))
      large=*(element+i);
  }
  return large;
}

void main()
{
  int n,*element;
  printf("Enter n: ");
  scanf("%d",&n);

  element=(int*)malloc(n*sizeof(int));
  for(int i=0;i<n;i++)
  {
    printf("Enter a element [%d] : ",i);
    scanf("%d",element+i);
  }
  printf("Largest element : %d\n",findLarge(element,n));
}
```

**Sample Input & Output:**

```
Enter n: 5
Enter a element [0] : 3
Enter a element [1] : 6
Enter a element [2] : 9
Enter a element [3] : 8
Enter a element [4] : 2
Largest element : 9
```

| Q3 | Write a program in C to Calculate the length of the string using a pointer. |
|---|---|

```c
#include <stdio.h>
int lenPtr(char *str)
{
  int i=0;
  while(*(str+i)!='\0')
  {
    i++;
  }
  return i;
}

void main()
{
  char *str;
  printf("Enter a String : ");
  scanf("%[^\n]",str);
  printf("Length of the String : %d\n",lenPtr(str));
}
```

**Sample Input & Output:**

```
Enter a String : hello world
Length of the String : 11
```

## Session 9 – More on Pointers – 05.12.2022

| Q1 | Use pointers to perform array sorting. |
|---|---|

```c
#include <stdio.h>

// Function to sort the numbers using pointers
void sort(int n, int* ptr)
{
  int i, j, t;

  // Sort the numbers using pointers
  for (i = 0; i < n; i++) {

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

      if (*(ptr + j) < *(ptr + i)) {

        t = *(ptr + i);
        *(ptr + i) = *(ptr + j);
        *(ptr + j) = t;
      }
    }
  }

    printf("\n");
  // print the numbers
  for (i = 0; i < n; i++)
    printf("%d ", *(ptr + i));
}

int main()
{
  int n;
  printf("Enter No. of elements: ");
  scanf("%d",&n);

  int arr[n];
    for(int i=0;i<n;i++)
    {
        printf("Enter element arr[%d] = ",i);
        scanf("%d",&arr[i]);
    }

  sort(n, arr);

  return 0;
}
```

**Sample Input & Output:**

```
Enter No. of elements: 5
Enter element arr[0] = 3
Enter element arr[1] = 6
Enter element arr[2] = 8
Enter element arr[3] = 1
Enter element arr[4] = 2

1 2 3 6 8
```

| Q2 | Explore how are 2D Arrays can be referenced with a pointer. Perform Simple input and output of matrix using pointers. |
|---|---|

```c
#include<stdio.h>

int main()
{
    int arr[3][4] = {
                        {11,22,33,44},
                        {55,66,77,88},
                        {11,66,77,44}
                    };

    int i, j;

    for(i = 0; i < 3; i++)
    {
        printf("Address of %d th array %p \n",i , *(arr + i));
        for(j = 0; j < 4; j++)
        {
             printf("arr[%d][%d]=%d\n", i, j, *( *(arr + i) + j) );
        }
        printf("\n\n");
    }

    return 0;
}
```

**Sample Input & Output:**

```
Address of 0 th array 0x7ffdd0c9f490
arr[0][0]=11
arr[0][1]=22
arr[0][2]=33
arr[0][3]=44


Address of 1 th array 0x7ffdd0c9f4a0
arr[1][0]=55
arr[1][1]=66
arr[1][2]=77
arr[1][3]=88


Address of 2 th array 0x7ffdd0c9f4b0
arr[2][0]=11
arr[2][1]=66
arr[2][2]=77
arr[2][3]=44
```

**Session 10 – Structures – 06.12.2022**

| | |
|---|---|
| Q1 | Create a datatype ‘fraction’ with numerator and denominator. Perform fraction addition, subtraction, multiplication, and division.<br>Note: Find LCM for addition and subtraction problems. Make it a menu-driven program. |

```c
#include <stdio.h>
struct fraction
{
  int num;
  int den;
};

int findGCD(int a,int b)
{
  if(a==0)
    return b;
  if(b==0)
    return a;
  if(a==b)
    return a;
  if(a>b)
    return findGCD(a-b,b);
  else
    return findGCD(a,b-a);
}

void add(struct fraction f1,struct fraction f2)
{
  int a,b,c,d,num,den,gcd;
  a=f1.num;
  b=f1.den;
  c=f2.num;
  d=f2.den;
  num=(a*d)+(b*c);
  den=b*d;
  gcd=findGCD(num,den);
  printf("Addition = %d/%d.\n\n",num/gcd,den/gcd);
}
void sub(struct fraction f1,struct fraction f2)
{
  int a,b,c,d,num,den,gcd;
  a=f1.num;
  b=f1.den;
  c=f2.num;
  d=f2.den;
  num=(a*d)-(b*c);
  den=b*d;
  gcd=findGCD(num,den);
  printf("Subtraction = %d/%d.\n\n",num/gcd,den/gcd);
```

```
}

void mul(struct fraction f1,struct fraction f2)
{
  int a,b,c,d,num,den,gcd;
  a=f1.num;
  b=f1.den;
  c=f2.num;
  d=f2.den;
  num=a*c;
  den=b*d;
  gcd=findGCD(num,den);
  printf("Multiplication = %d/%d.\n\n",num/gcd,den/gcd);
}

void div(struct fraction f1,struct fraction f2)
{
  int a,b,c,d,num,den,gcd;
  a=f1.num;
  b=f1.den;
  c=f2.num;
  d=f2.den;
  num=a*d;
  den=b*c;
  gcd=findGCD(num,den);
  printf("Division = %d/%d.\n\n",num/gcd,den/gcd);
}

void main()
{
  struct fraction f1,f2;
  int n,d,op;
  printf("\nEnter a numerator 1= ");
  scanf("%d",&f1.num);
  printf("Enter a denominator 1= ");
  scanf("%d",&f1.den);
  printf("\nEnter a numerator 2= ");
  scanf("%d",&f2.num);
  printf("Enter a denominator 2= ");
  scanf("%d",&f2.den);

  printf("\nMENU:\n1.Addition 2.Subtraction 3.Multiply 4.Division 5.Exit\n");
  printf("\nEnter a Operation : ");
  scanf("%d",&op);

  do{
    switch(op)
    {
      case 1:
        add(f1,f2);break;
      case 2:
        sub(f1,f2);break;
```

```
      case 3:
        mul(f1,f2);break;
      case 4:
        div(f1,f2);break;
    }
    printf("Enter a Operation : ");
    scanf("%d",&op);
  }while(op!=5);
  printf("Thank you!!!\n\n");
}
```

**Sample Input & Output:**

```
Enter a numerator 1= 7
Enter a denominator 1= 6

Enter a numerator 2= 2
Enter a denominator 2= 3

MENU:
1.Addition 2.Subtraction 3.Multiply 4.Division 5.Exit

Enter a Operation : 1
Addition = 11/6.

Enter a Operation : 2
Subtraction = 1/2.

Enter a Operation : 3
Multiplication = 7/9.

Enter a Operation : 4
Division = 7/4.

Enter a Operation : 5
Thank you!!!
```

| | |
|---|---|
| Q2 | Create a datatype ‘Student’ with name, roll number, and marks for 5 subjects as an array. Maintain an array of student type and sort them based on their total marks in descending order. |

```
#include<stdio.h>
struct Student
{
  char name[10];
  int rno;
  int mark[5];
  int total;
};
```

```c
void main()
{
  int n,i,j,total[5];
  struct Student s[5],temp;
  printf("Enter a no.of Student : ");
  scanf("%d",&n);

  //Read Input:
  for(i=0;i<n;i++)
  {
    int sum=0;
    printf("\nStudent %d :-",i+1);
    printf("\nEnter Name :- ");
    scanf(" %[^\n]",s[i].name);
    printf("Enter Roll Number:- ");
    scanf("%d",&s[i].rno);
    s[i].total=0;
    for(j=0;j<5;j++)
    {
      printf("Enter mark %d :- ",j+1);
      scanf("%d",&s[i].mark[j]);
      sum+=s[i].mark[j];
    }
    s[i].total+=sum;
  }


  for(i=0;i<n;i++)
  {
    for(j=i+1;j<n;j++)
    {
      if(s[i].total<s[j].total)
      {
        temp=s[i];
        s[i]=s[j];
        s[j]=temp;
      }
    }
  }

  printf("\n\n----------------------------");
  printf("\n     Student Record        ");
  printf("\n----------------------------");
  for(int i=0;i<n;i++)
  {
    printf("\n\nStudent %d : \n",i+1);
    printf("Name: %s\nRollno :%d\nTotal Mark : %d
",s[i].name,s[i].rno,s[i].total);
  }
}
```

**Sample Input & Output:**

```
Enter a no.of Student : 3

Student 1 :-
Enter Name :- Aarav
Enter Roll Number:- 1
Enter mark 1 :- 62
Enter mark 2 :- 75
Enter mark 3 :- 77
Enter mark 4 :- 68
Enter mark 5 :- 88

Student 2 :-
Enter Name :- Bhavin
Enter Roll Number:- 2
Enter mark 1 :- 85
Enter mark 2 :- 85
Enter mark 3 :- 92
Enter mark 4 :- 89
Enter mark 5 :- 93

Student 3 :-
Enter Name :- Sanjay
Enter Roll Number:- 3
Enter mark 1 :- 77
Enter mark 2 :- 55
Enter mark 3 :- 90
Enter mark 4 :- 81
Enter mark 5 :- 55
```

```
--------------------------
     Student Record
--------------------------

Student 1 :
Name: Bhavin
Rollno :2
Total Mark : 444

Student 2 :
Name: Aarav
Rollno :1
Total Mark : 370

Student 3 :
Name: Sanjay
Rollno :3
Total Mark : 358
```

**Session 12 – Pointers & Structures – 08.12.2022**

| | |
|---|---|
| Q1 | Implement singly linked list in C using structures and pointers with the following functionalities<br>. Insert front<br>. Insert end<br>. Insert in middle<br>. Delete a node<br>. Display the list |

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
```

```
struct node *head;

void beginsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{
    int choice =0;
    while(choice != 9)
    {
        printf("\n\n*********Main Menu*********\n");
        printf("\nChoose one option from the following list ...\n");
        printf("\n===============================================\n");
        printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any
random location\n4.Delete from Beginning\n5.Delete from last\n6.Delete node
after specified location\n7.Search for an element\n8.Show\n9.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            beginsert();
            break;
            case 2:
            lastinsert();
            break;
            case 3:
            randominsert();
            break;
            case 4:
            begin_delete();
            break;
            case 5:
            last_delete();
            break;
            case 6:
            random_delete();
            break;
            case 7:
            search();
            break;
            case 8:
            display();
            break;
            case 9:
            exit(0);
            break;
```

```
            default:
            printf("Please enter valid choice..");
        }
    }
}
void beginsert()
{
    struct node *ptr;
    int item;
    ptr = (struct node *) malloc(sizeof(struct node *));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter value\n");
        scanf("%d",&item);
        ptr->data = item;
        ptr->next = head;
        head = ptr;
        printf("\nNode inserted");
    }

}
void lastinsert()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter value?\n");
        scanf("%d",&item);
        ptr->data = item;
        if(head == NULL)
        {
            ptr -> next = NULL;
            head = ptr;
            printf("\nNode inserted");
        }
        else
        {
            temp = head;
            while (temp -> next != NULL)
            {
                temp = temp -> next;
            }
```

```
            temp->next = ptr;
            ptr->next = NULL;
            printf("\nNode inserted");

        }
    }
}
void randominsert()
{
    int i,loc,item;
    struct node *ptr, *temp;
    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter element value");
        scanf("%d",&item);
        ptr->data = item;
        printf("\nEnter the location after which you want to insert ");
        scanf("\n%d",&loc);
        temp=head;
        for(i=0;i<loc;i++)
        {
            temp = temp->next;
            if(temp == NULL)
            {
                printf("\ncan't insert\n");
                return;
            }

        }
        ptr ->next = temp ->next;
        temp ->next = ptr;
        printf("\nNode inserted");
    }
}
void begin_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nList is empty\n");
    }
    else
    {
        ptr = head;
        head = ptr->next;
        free(ptr);
        printf("\nNode deleted from the begining ...\n");
```

```
    }
}
void last_delete()
{
    struct node *ptr,*ptr1;
    if(head == NULL)
    {
        printf("\nlist is empty");
    }
    else if(head -> next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nOnly node of the list deleted ...\n");
    }

    else
    {
        ptr = head;
        while(ptr->next != NULL)
        {
            ptr1 = ptr;
            ptr = ptr ->next;
        }
        ptr1->next = NULL;
        free(ptr);
        printf("\nDeleted Node from the last ...\n");
    }
}
void random_delete()
{
    struct node *ptr,*ptr1;
    int loc,i;
    printf("\n Enter the location of the node after which you want to perform
deletion \n");
    scanf("%d",&loc);
    ptr=head;
    for(i=0;i<loc;i++)
    {
        ptr1 = ptr;
        ptr = ptr->next;

        if(ptr == NULL)
        {
            printf("\nCan't delete");
            return;
        }
    }
    ptr1 ->next = ptr ->next;
    free(ptr);
    printf("\nDeleted node %d ",loc+1);
}
```

```c
void search()
{
    struct node *ptr;
    int item,i=0,flag;
    ptr = head;
    if(ptr == NULL)
    {
        printf("\nEmpty List\n");
    }
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        while (ptr!=NULL)
        {
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
        if(flag==1)
        {
            printf("Item not found\n");
        }
    }

}

void display()
{
    struct node *ptr;
    ptr = head;
    if(ptr == NULL)
    {
        printf("Nothing to print");
    }
    else
    {
        printf("\nprinting values . . . . .\n");
        while (ptr!=NULL)
        {
            printf("\n%d",ptr->data);
            ptr = ptr -> next;
        }
    }}
```