#### **DEPARTMENT OF COMPUTER SCIENCE**

#### **RAJAGIRI COLLEGE OF SOCIAL SCIENCES**

#### **(Autonomous)**

**KALAMASSERY - KOCHI - 683104**

****

### MASTER OF COMPUTER APPLICATIONS

### DATA STRUCTRES

### LAB RECORD

**NAME : MUHAMMAD ANSHAD P A**

**SEMESTER : FIRST Semester**

**REGISTER NO. : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#### **Logo**

#### **DEPARTMENT OF COMPUTER SCIENCE**

#### **RAJAGIRI COLLEGE OF SOCIAL SCIENCES**

#### **(Autonomous)**

KALAMASSERY - KOCHI – 683104

### MASTER OF COMPUTER APPLICATIONS

###### CERTIFICATE

NAME : **MUHAMMAD ANSHAD P A**

**SEMESTER : FIRST Semester**

REGISTER NO. : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Certified that this is a bonafide record of work done by the student in the Software Laboratory of Rajagiri Department of Computer Science, Kalamassery.

Faculty in Charge Dean, Computer Science

Internal Examiner External Examiner

Place : Kalamassery

Date :

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Program**  **NO.** | **Description** | **PAGE**  **NO.** |
| **1\_1** | Write programs to demonstrate the use of storage classes in C. | **1** |
| **1\_2** | Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use global variables) | **3** |
| **1\_3** | Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use only local variables) | **8** |
| **1\_4** | Search for all the occurrences of an element in an integer array (positions) | **12** |
| **1\_5** | Sort the array elements in ascending order (minimum three functions: read, disp and sort) | **14** |
| **1\_6** | Two-dimensional matrix: using functions   1. Addition 2. Subtraction 3. Multiplication 4. Transpose 5. Determinant | **16** |
| **1\_7** | Display the array elements in the same order using a recursive function | **21** |
| **1\_8** | Display array elements in reverse order using a recursive function | **23** |
| **2\_1** | Implement stack operations using arrays. | **25** |
| **2\_2** | Reverse a string using Stack | **29** |
| **3\_1** | Convert an expression from infix to postfix using stack | **30** |
| **3\_2** | Evaluate an expression using stack | **32** |
| **4\_1** | Define a structure for dates with dd/mm/yyyy. Provide functions for reading, displaying and comparing two dates are equal or not | **35** |
| **4\_2** | Define a structure for employees with eno,ename, esal and dno. Read n employees information and provide functions for the following:   1. Searching an employee by no 2. Sorting the employees by   Name  Salary   1. Deleting an employee | **37** |
| **5\_1** | Read a polynomial and display it using array | **45** |
| **5\_2** | Add two polynomials using the array itself | **48** |
| **6\_1** | Read a polynomial and display it using structure array | **52** |
| **6\_2** | Add two polynomials | **55** |
| **6\_3** | Subtract two polynomials | **59** |
| **6\_4** | Multiply two polynomials | **64** |
| **7\_1** | Implement a) malloc , b) calloc and c) free functions | **68** |
| **7\_2** | Use malloc to read n integers and find the mean. | **70** |
| **7\_3** | Use calloc to read n numbers and find the mode. | **72** |
| **7\_4** | Declare a structure for Books having author\_name and book\_name. Create an array of books using a pointer variable. Provide functions for reading n books and displaying the same using pointers. | **74** |
| **7\_5** | Use realloc to implement varchar for any length. | **77** |
| **8\_1** | Implement Queue using array | **78** |
| **8\_2** | Implement priority queue | **82** |
| **9\_1** | Demonstrate a linked list creation and display | **85** |
| **9\_2** | Write a program with functions  to insert a new node  at the beginning of a Singly Linked List.  At the end of the linked list  after a specified element in a linked list. | **88** |
| **9\_3** | Write a program with functions to delete a nodeFrom the beginning of the linked list  From the end of the linked list  The node with specified data element | **93** |
| **9\_4** | Write a program to create a singly linked list of n nodes and display it in reverse order. | **100** |
| **9\_5** | Sort the elements in a linked list using  changing the values (swapping the values)  Changing the address (Swapping the address) | **104** |
| **10\_1** | Polynomial using linked list - addition and multiplication | **110** |
| **10\_2** | Linked list using names - insert, delete, display, sort, reverse, count | **115** |
| **11\_1\_a** | Linked Stack | **122** |
| **11\_1\_b** | Linked Queue | **126** |
| **11\_1\_c** | Circular Linked List | **130** |
| **11\_1\_d** | Circular Linked Queue | **135** |
| **11\_1\_e** | Doubly Linked List | **139** |
| **11\_1\_f** | Circular doubly linked list - store string values as data part | **143** |
| **12\_1** | Binary search tree insertion and display Traversal using inorder, preorder and postorder using recursion. | **147** |
| **12\_2** | Binary search tree insertion and display in-order without using recursion | **151** |
| **12\_3** | Binary search tree insertion and display pre-order without using recursion | **156** |
| **12\_4** | Binary search tree insertion and display post-order without using recursion | **161** |
| **12\_5** | Binary search tree insertion using names and display the names in ascending order using inorder traversal. | **166** |
| **13\_1** | Demonstrate the data structure of adjacent matrix  using arrays | **169** |
| **13\_2** | Demonstrate the data structure of adjacent matrix  using linked lists | **172** |

|  |  |
| --- | --- |
| **Program #1\_1** | **Date: 10/08/2022** |
| **Write a program to demonstrate the use of storage classes in C.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_1 : PROGRAM TO DEMONSTRATE THE USE OF STORAGE CLASSES IN C.  #include <stdio.h>  int gv1,gv2=3;//golbal variable  void auto\_var()  {      printf("\n-Auto variable-\n");      auto int au = 36;//auto variable      printf("auto variable au = %d\n",au);  }  void static\_var()  {      static int i,k=4; //static variable      printf("\n-Static variable-\n static value of i = %d\n",i);      printf("static value of k = %d\n",k++);  }  void register\_var()  {      int i=10;      register int \*m = &i; //register variable      printf("\nValue of regsiter variable  = %d\n",\*m);  }  int main()  {      int y,z=20;//z is local variable to main()      printf("\n-Local variable-\nValues of y = %d and z = %d",y,z);      auto\_var();      static\_var();      static\_var();      static\_var();      register\_var();      printf("\n-Global variable-\nValues of gv1 = %d and gv2 = %d",gv1,gv2);        return 0;    }  //extern storage class main file :  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //#include <stdio.h>  //int count ;  //extern void write\_extern();  //int main() {  //count = 5;  //write\_extern();  //}  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //extern second file :  //#include <stdio.h>    //extern int count;    //void write\_extern(void) {  //printf("count is %d\n", count);  //}  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //OUTPUT : count is 5 |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_1\_demo\_storage\_classes.exe'  -Local variable-  Values of y = 2338816 and z = 20  -Auto variable-  auto variable au = 36  -Static variable-  static value of i = 0  static value of k = 4  -Static variable-  static value of i = 0  static value of k = 5  -Static variable-  static value of i = 0  static value of k = 6  Value of regsiter variable = 10  -Global variable-  Values of gv1 = 0 and gv2 = 3  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #1\_2** | **Date: 12/08/2022** |
| **Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use global variables)** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_2 : USE A MENU-DRIVEN PROGRAM TO INSERT, SEARCH, DELETE AND SORT ELEMENTS IN AN ARRAY USING FUNCTIONS (USE GLOBAL VARIABLES)  #include <stdio.h>  #define MAX\_SIZE 100  //Global variables  int arr[MAX\_SIZE];  int size=0,i=0,j=0;  //Declaring Functions  void menu();  void insertEl();  void searchEl();  void deleteEl();  void sortArr();  void displayArr();  int main(){      int choice;      do{          menu();          printf("\nEnter your choice : ");          scanf("%d",&choice);            switch(choice){              case 1:                  insertEl();                  break;              case 2:                  searchEl();                  break;              case 3:                  deleteEl();                  break;              case 4:                  sortArr();                  break;              case 5:                  displayArr();                  break;              case 6:                  printf("\nExiting program...\n");                  break;              default:                  printf("inavlid input\n");                  break;              }        }while(choice!=6);  }  void menu(){      printf("\n----MENU----\n");      printf("1.Insert.\n2.Search.\n3.Delete.\n4.Sort.\n5.Display.\n6.Exit.\n");  }  void insertEl(){      int trv=0;      printf("\nEnter how much elements to be inserted : ");      scanf("%d",&trv);      for(i=0;i<trv;i++)      {          if(size >= MAX\_SIZE)          {              printf("Array is full.Cannot insert more elements\n");          }          else{              printf("Enter the element to be inserted : ");              scanf("%d",&arr[size]);              size++;          }      }    }  void searchEl(){      int src,found=0;      if(size==0){          printf("Array is empty.Cannot perform search.\n");      }      printf("Enter the element to search : ");      scanf("%d",&src);        for(i=0;i<size;i++)      {          if(arr[i]==src)          {              printf("Element %d found at index %d.\n",src,i);              found=1;          }      }      if(!found){          printf("Element not found .\n");      }  }  void deleteEl(){      int del,found=0;        if(size==0)      {          printf("Array is empty.Cannot perform deleteion");        }      else{          printf("Enter the element to be deleted : ");          scanf("%d",&del);            for(i=0;i<size;i++){              if(arr[i]==del){                  found=1;                  break;              }          }          if(found==1){              //moving elements              for(j=i;j<size-1;j++){                  arr[j]=arr[j+1];              }size--;              printf("Element %d deleted.\n",del);            }else{              printf("Element not found in array.Deletion failed\n");          }      }    }  void sortArr(){      int temp;        for(i=0;i<size-1;i++)      {          for(j=i+1;j<size;j++){              if(arr[i]>arr[j]){                  temp=arr[i];                  arr[i]=arr[j];                  arr[j]=temp;              }          }      }      printf("Array sorted in ascending order .\n");  }  void displayArr(){      if(size==0)      {          printf("Array is empty.\n");        }      printf("Array elements :- \n");      for(i=0;i<size;i++){          printf("%d ",arr[i]);      }      printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_2\_menudriven\_insert\_sort\_GV.exe'  ----MENU----  1.Insert.  2.Search.  3.Delete.  4.Sort.  5.Display.  6.Exit.  Enter your choice : 1  Enter how much elements to be inserted : 5  Enter the element to be inserted : 65  Enter the element to be inserted : 12  Enter the element to be inserted : 5  Enter the element to be inserted : 2  Enter the element to be inserted : 20  ----MENU----  1.Insert.  2.Search.  3.Delete.  4.Sort.  5.Display.  6.Exit.  Enter your choice : 2  Enter the element to search : 5  Element 5 found at index 2.  ----MENU----  1.Insert.  2.Search.  3.Delete.  4.Sort.  5.Display.  6.Exit.  Enter your choice : 3  Enter the element to be deleted : 5  Element 5 deleted.  ----MENU----  1.Insert.  2.Search.  3.Delete.  4.Sort.  5.Display.  6.Exit.  Enter your choice : 4  Array sorted in ascending order .  ----MENU----  1.Insert.  2.Search.  3.Delete.  4.Sort.  5.Display.  6.Exit.  Enter your choice : 5  Array elements :-  2 12 20 65  ----MENU----  1.Insert.  2.Search.  3.Delete.  4.Sort.  5.Display.  6.Exit.  Enter your choice : 6  Exiting program...  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #1\_3** | **Date: 15/08/2022** |
| **Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use only local variables)** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_3 : USE A MENU-DRIVEN PROGRAM TO INSERT, SEARCH, DELETE AND SORT ELEMENTS IN AN ARRAY USING FUNCTIONS (USE ONLY LOCAL VARIABLES)  #include <stdio.h>  #define MAX\_SIZE 100  void insertEl(int arr[], int \*size) {      int trv=0,i,element;      printf("\nEnter how much elements to be inserted : ");      scanf("%d",&trv);      for(i=0;i<trv;i++)      {          if (\*size < MAX\_SIZE) {              printf("Enter element to insert: ");              scanf("%d", &element);              arr[(\*size)++] = element;              printf("Element inserted successfully.\n");          } else {              printf("Array is full. Cannot insert more elements.\n");          }      }  }  void searchEl(const int arr[], int size, int element) {      int found = 0,i;      for (i = 0; i < size; i++) {          if (arr[i] == element) {              printf("Element %d found at index %d.\n", element, i);              found = 1;              break;          }      }      if (!found) {          printf("Element %d not found in the array.\n", element);      }  }  void deleteEl(int arr[], int \*size, int element) {      int found = 0,i,j;      for (i = 0; i < \*size; i++) {          if (arr[i] == element) {              found = 1;              for (j = i; j < \*size - 1; j++) {                  arr[j] = arr[j + 1];              }              (\*size)--;              printf("Element %d deleted successfully.\n", element);              break;          }      }      if (!found) {          printf("Element %d not found in the array. Deletion failed.\n", element);      }  }  void sortArr(int arr[], int size) {      int i,j;      for (i = 0; i < size - 1; i++) {          for (j = 0; j < size - i - 1; j++) {              if (arr[j] > arr[j + 1]) {                  int temp = arr[j];                  arr[j] = arr[j + 1];                  arr[j + 1] = temp;              }          }      }      printf("Array sorted successfully.\n");  }  void displayArr(int arr[],int size){      int i;      if(size==0)      {          printf("Array is empty.\n");        }      printf("Array elements :- \n");      for(i=0;i<size;i++){          printf("%d ",arr[i]);      }      printf("\n");  }  int main() {      int arr[MAX\_SIZE];      int size = 0;      int choice, element;        do {          printf("\n----Menu----\n1. Insert element\n2. Search element\n3. Delete element\n4. Sort array\n5. Display.\n6. Exit\n");          printf("Enter your choice: ");          scanf("%d", &choice);            switch (choice) {              case 1:                  insertEl(arr, &size);                  break;              case 2:                  printf("Enter element to search: ");                  scanf("%d", &element);                  searchEl(arr, size, element);                  break;              case 3:                  printf("Enter element to delete: ");                  scanf("%d", &element);                  deleteEl(arr, &size, element);                  break;              case 4:                  sortArr(arr, size);                  break;              case 5:                  displayArr(arr, size);                  break;                case 6:                  printf("Exiting...\n");                  break;              default:                  printf("Invalid choice. Please enter a valid option.\n");          }      } while (choice != 5);        return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_3\_menudriven\_insert\_sort\_driven\_insert\_sort\_LC.exe'  ----Menu----  1. Insert element  2. Search element  3. Delete element  4. Sort array  5. Display.  6. Exit  Enter your choice: 1  Enter how much elements to be inserted : 5  Enter element to insert: 65  Enter element to insert: 12  Enter element to insert: 5  Enter element to insert: 2  Enter element to insert: 20  ----Menu----  1. Insert element  2. Search element  3. Delete element  4. Sort array  5. Display.  6. Exit  Enter your choice: 2  Enter element to search: 5  Element 5 found at index 2.  ----Menu----  1. Insert element  2. Search element  3. Delete element  4. Sort array  5. Display.  6. Exit  Enter your choice: 3  Enter element to delete: 5  Element 5 deleted successfully.  ----Menu----  1. Insert element  2. Search element  3. Delete element  4. Sort array  5. Display.  6. Exit  Enter your choice: 4  Array sorted successfully.  ----Menu----  1. Insert element  2. Search element  3. Delete element  4. Sort array  5. Display.  6. Exit  Enter your choice: 5  Array elements :-  2 12 20 65  ----Menu----  1. Insert element  2. Search element  3. Delete element  4. Sort array  5. Display.  6. Exit  Enter your choice: 6  Exiting...  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #1\_4** | **Date: 15/08/2022** |
| **Search for all the occurrences of an element in an integer array (positions).** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_4 : SEARCH FOR ALL OCCURENCES OF AN ELEMENT IN AN INTEGER ARRAY  #include <stdio.h>  #define MAX\_SIZE 100  int i,arr[MAX\_SIZE];  void findOcc(const int arr[], int size, int element) {      int found = 0;        printf("Occurrences of %d at positions: ", element);        for (i = 0; i < size; i++) {          if (arr[i] == element) {              printf("%d ", i);              found = 1;          }      }        if (!found) {          printf("None");      }        printf("\n");  }  int main() {        int size, element;        printf("Enter the size of the array: ");      scanf("%d", &size);        if (size <= 0 || size > MAX\_SIZE) {          printf("Invalid array size.\n");        }        printf("\nEnter the elements of the array : \n");      for (i = 0; i < size; i++) {          scanf("%d", &arr[i]);      }        printf("\nEnter the element to search for : ");      scanf("%d", &element);        findOcc(arr, size, element);        return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE> cd 'e:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output'  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_4\_search\_for\_occurance.exe'  Enter the size of the array: 5  Enter the elements of the array :  20  42  76  20  20  Enter the element to search for : 20  Occurrences of 20 at positions: 0 3 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #1\_5** | **Date: 16/08/2022** |
| **Sort the array elements in ascending order (minimum three functions: read, disp and sort).** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_5 : SORT THE ARRAY ELEMENTS IN  ASCENDING ORDER(min three functions:read,disp,sort)  #include <stdio.h>  int i,j;  //Read array elements  void read(int arr[], int size) {        printf("Enter %d elements:\n", size);      for (i = 0; i < size; i++) {          scanf("%d", &arr[i]);      }  }  //Display array elements  void disp(int arr[], int size) {      printf("Array elements: ");      for (i = 0; i < size; i++) {          printf("%d ", arr[i]);      }      printf("\n");  }  //Sort array elements in asc order - Bubble Sort  void sort(int arr[], int size) {      for (i = 0; i < size - 1; i++) {          for (j = 0; j < size - i - 1; j++) {              if (arr[j] > arr[j + 1]) {                  // Swap arr[j] and arr[j+1]                  int temp = arr[j];                  arr[j] = arr[j + 1];                  arr[j + 1] = temp;              }          }      }  }  int main() {      int size;      printf("Enter the size of the array: ");      scanf("%d", &size);      int arr[size];      read(arr, size);      printf("Before sorting:\n");      disp(arr, size);      sort(arr, size);      printf("After sorting in ascending order:\n");      disp(arr, size);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE> cd 'e:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output'  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_5\_sort\_array\_asc.exe'  Enter the size of the array: 6  Enter 6 elements:  25  15  2  19  60  20  Before sorting:  Array elements: 25 15 2 19 60 20  After sorting in ascending order:  Array elements: 2 15 19 20 25 60  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |
| --- |
| **Program #1\_6** |
| **Two-dimensional matrix: using functions**   * 1. **Addition**   2. **Subtraction**   3. **Multiplication**   4. **Transpose**   5. **Determinant** |

**Source Code:**

|  |
| --- |
| /\*PROGRAM 1\_6 : TWO DIMENTIONAL MATRIX USING FUNCTIONS:  a.Addition  b.Subtraction  c.Multiplication  \*/  #include <stdio.h>  #define MAX\_SIZE 10  int i,j,k;  void inputMat(int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {      printf("Enter matrix elements:\n");      for (i = 0; i < rows; i++) {          for (j = 0; j < cols; j++) {              scanf("%d", &matrix[i][j]);          }      }  }  void dispMat(int matrix[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {      printf("Matrix:\n");      for (i = 0; i < rows; i++) {          for (j = 0; j < cols; j++) {              printf("%d ", matrix[i][j]);          }          printf("\n");      }  }  void addMat(int mat1[MAX\_SIZE][MAX\_SIZE], int mat2[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {      for (i = 0; i < rows; i++) {          for (j = 0; j < cols; j++) {              result[i][j] = mat1[i][j] + mat2[i][j];          }      }  }  void subMat(int mat1[MAX\_SIZE][MAX\_SIZE], int mat2[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {      for (i = 0; i < rows; i++) {          for (j = 0; j < cols; j++) {              result[i][j] = mat1[i][j] - mat2[i][j];          }      }  }  void multMat(int mat1[MAX\_SIZE][MAX\_SIZE], int mat2[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE], int rows1, int cols1, int cols2) {        for (i = 0; i < rows1; i++) {          for (j = 0; j < cols2; j++) {              result[i][j] = 0;              for (k = 0; k < cols1; k++) {                  result[i][j] += mat1[i][k] \* mat2[k][j];              }          }      }  }  int determinant(int matrix[MAX\_SIZE][MAX\_SIZE], int n) {      int det = 0;      if (n == 1) {          return matrix[0][0];      } else if (n == 2) {          return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];      } else {          for (k = 0; k < n; k++) {              int submatrix[MAX\_SIZE][MAX\_SIZE];              int subi = 0;              for (i = 1; i < n; i++) {                  int subj = 0;                  for (j = 0; j < n; j++) {                      if (j == k) continue;                      submatrix[subi][subj] = matrix[i][j];                      subj++;                  }                  subi++;              }              det += (k % 2 == 0 ? 1 : -1) \* matrix[0][k] \* determinant(submatrix, n - 1);          }          return det;      }  }  void transMat(int matrix[MAX\_SIZE][MAX\_SIZE], int result[MAX\_SIZE][MAX\_SIZE], int rows, int cols) {      for (i = 0; i < rows; i++) {          for (j = 0; j < cols; j++) {              result[j][i] = matrix[i][j];          }      }  }  int main() {      int mat1[MAX\_SIZE][MAX\_SIZE], mat2[MAX\_SIZE][MAX\_SIZE], result[MAX\_SIZE][MAX\_SIZE];      int rows1, cols1, rows2, cols2;      printf("Enter the number of rows and columns for matrix 1: ");      scanf("%d %d", &rows1, &cols1);      inputMat(mat1, rows1, cols1);      printf("Enter the number of rows and columns for matrix 2: ");      scanf("%d %d", &rows2, &cols2);      inputMat(mat2, rows2, cols2);      if (rows1 != rows2 || cols1 != cols2) {          printf("\nMatrix addition, subtraction, and multiplication are not possible.\n");      } else {          // Addition          addMat(mat1, mat2, result, rows1, cols1);          printf("\nMatrix Addition:\n");          dispMat(result, rows1, cols1);          // Subtraction          subMat(mat1, mat2, result, rows1, cols1);          printf("\nMatrix Subtraction:\n");          dispMat(result, rows1, cols1);          // Multiplication          multMat(mat1, mat2, result, rows1, cols1, cols2);          printf("\nMatrix Multiplication:\n");          dispMat(result, rows1, cols2);      }      // Determinant      if (rows1 == cols1) {          int det = determinant(mat1, rows1);          printf("\n\nDeterminant of matrix 1: %d\n", det);      } else {          printf("Determinant can only be calculated for square matrices.\n");      }          if (rows2 == cols2) {          int det = determinant(mat2, rows2);          printf("\nDeterminant of matrix 2: %d\n", det);      } else {          printf("Determinant can only be calculated for square matrices.\n");      }      // Transpose      int transpose[MAX\_SIZE][MAX\_SIZE];      transMat(mat1, transpose, rows1, cols1);      printf("\n\nTranspose of matrix 1:\n");      dispMat(transpose, cols1, rows1);      transMat(mat2, transpose, rows2, cols2);      printf("\nTranspose of matrix 1:\n");      dispMat(transpose, cols2, rows2);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_6\_two\_dimens\_matrix\_addition.exe'  Enter the number of rows and columns for matrix 1: 2  2  Enter matrix elements:  1  2  3  4  Enter the number of rows and columns for matrix 2: 2  2  Enter matrix elements:  1  0  2  1  Matrix Addition:  Matrix:  2 2  5 5  Matrix Subtraction:  Matrix:  0 2  1 3  Matrix Multiplication:  Matrix:  5 2  11 4  Determinant of matrix 1: -2  Determinant of matrix 2: 1  Transpose of matrix 1:  Matrix:  1 3  2 4  Transpose of matrix 1:  Matrix:  1 2  0 1  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #1\_7** | **Date: 18/08/2022** |
| **Display the array elements in the same order using a recursive function .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_7: DISPLAY ARRAY ELEMENTS IN THE SAME ORDER USING A RECURSIVE FUNCTION  #include <stdio.h>  void dispRecc(int arr[], int size, int index) {      if (index == size) {          return;      }        printf("%d ", arr[index]);      dispRecc(arr, size, index + 1); // Recursively calling  for the next index  }  int main() {      int size,i;        printf("Enter the size of the array: ");      scanf("%d", &size);        int arr[size];        printf("Enter array elements:\n");      for (i = 0; i < size; i++) {          scanf("%d", &arr[i]);      }        printf("Array elements in the same order: ");      dispRecc(arr, size, 0);        return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE> cd 'e:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output'  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_7\_display\_array\_using\_reccursive\_funct.exe'  Enter the size of the array: 5  Enter array elements:  25  62  75  101  12  Array elements in the same order: 25 62 75 101 12  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #1\_8** | **Date: 18/08/2022** |
| **Display array elements in reverse order using a recursive function .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 1\_8: DISPALY ARRAY ELEMENTS IN REVERSE ORDER USING A RECURSIVE FUNCTION  #include <stdio.h>  void dispRevRec(int arr[], int size, int index) {      if (index < 0) {          return;      }        printf("%d ", arr[index]);      dispRevRec(arr, size, index - 1); // Recursively calling for the previous index  }  int main() {      int size,i;        printf("Enter the size of the array: ");      scanf("%d", &size);        int arr[size];        printf("Enter array elements:\n");      for (i = 0; i < size; i++) {          scanf("%d", &arr[i]);      }        printf("Array elements in reverse order: ");      dispRevRec(arr, size, size - 1); // Start displaying from the last index        return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE> cd 'e:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output'  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'1\_8\_reccursive\_reverse\_disp.exe'  Enter the size of the array: 5  Enter array elements:  10  20  30  40  50  Array elements in reverse order: 50 40 30 20 10  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #2\_1** | **Date: 19/08/2022** |
| **Implement stack operations using arrays.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 2\_1: IMPLEMENT STACK OPERATIONS USING ARRAY  #include <stdio.h>  #define MAX\_SIZE 10  int stack[MAX\_SIZE];  int top=-1;  int menu()  {      int choice;      printf("\n\_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_\n");      printf("1.PUSH\n2.POP\n3.PEEK\n4.Exit\n");      printf("\nEnter your choice : ");      scanf("%d",&choice);      return choice;  }  void push(int val)  {      if(top==MAX\_SIZE-1)      {          printf("\nStack Overflow.Cannot PUSH %d \n",&val);      }      else      {          stack[++top]=val;          printf("\n%d is pushed to stack \n",val);      }    }  void pop()  {      if(top== -1)      {          printf("\nStack Underflow.Cannot perform POP\n");      }      else      {          printf("\n%d Popped from stack.\n",stack[top--]);      }  }  void peek()  {      if(top== -1)      {          printf("\nStack Underflow.Cannot perform PEEK\n");      }      else      {          printf("\nTop of stack is : %d\n",stack[top]);      }  }  int main()  {      int choice,val,i;      for(choice=menu();choice!=4;choice=menu())      {              switch(choice)          {              case 1:                  printf("\nEnter the value to PUSH : ");                  scanf("%d",&val);                  push(val);                  break;              case 2:                  pop();                  break;              case 3:                  peek();                  break;              case 4:                  printf("\nExiting......\n");                  break;              default:                  printf("\nInvalid choice.please enter valid choice");            }      }      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'2\_9\_stack\_operations.exe'  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 2  Stack Underflow.Cannot perform POP  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 1  Enter the value to PUSH : 10  10 is pushed to stack  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 1  Enter the value to PUSH : 20  20 is pushed to stack  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 1  Enter the value to PUSH : 30  30 is pushed to stack  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 2  30 Popped from stack.  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 3  Top of stack is : 20  \_\_\_\_\_\_\_\_\_Stack Operations\_\_\_\_\_\_\_\_\_  1.PUSH  2.POP  3.PEEK  4.Exit  Enter your choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #2\_2** | **Date: 10/08/2022** |
| **Reverse a string using Stack .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 2\_2:REVERSE A STRING USING STACK  #include <stdio.h>  #include <string.h>  #define MAX\_SIZE 100  char stack[MAX\_SIZE],top=-1;  char input[MAX\_SIZE];  void push(char input[20])  {      int i,len=strlen(input);        for(i=0;i<len;i++)      {          top++;          stack[top]=input[i];      }    }  void pop(){      printf("\nReversed string is to reverse it : ");      while(top>=-1)      {          printf("%c",stack[top]);          top--;      }  }  void main(){      printf("\nEnter the string : ");      scanf("%s",&input);      push(input);      pop();  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'2\_10\_reverse\_string\_stack.exe'  Enter the string : anshad  Reversed string is to reverse it : dahsna  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #3\_1** | **Date: 21/08/2022** |
| **Convert an expression from infix to postfix using stack .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 3\_1 : CONVERT AN EXPRESSION FROM INFIX TO POSTFIX USING STACK  #include <stdio.h>  #include <string.h>  #define MAX 10  char stk[MAX];  int top = -1;  void push(char x) {      top++;      stk[top] = x;  }  char pop() {      char y = stk[top];      top--;      return y;  }  int precedence(char k) {      if (k == '^') {          return 3;      } else if (k == '\*' || k == '/') {          return 2;      } else if (k == '+' || k == '-') {          return 1;      } else {          return 0;      }  }  void conversion() {      char infix[MAX], postfix[MAX];      printf("Enter infix expression: ");      scanf("%s", infix);        int i = 0, j = 0;      char temp,k;      while (infix[i] != '\0') {          temp = infix[i];          switch (temp) {              case '(':                  push(temp);                  break;              case ')':                   k = pop();                  while (k != '(') {                      postfix[j] = k;                      j++;                      k = pop();                  }                  break;              case '^':              case '\*':              case '/':              case '+':              case '-':                  while (precedence(stk[top]) >= precedence(temp)) {                      postfix[j] = pop();                      j++;                  }                  push(temp);                  break;              default:                  postfix[j] = temp;                  j++;          }          i++;      }      while (top > -1) {          postfix[j] = pop();          j++;      }      postfix[j] = '\0';      printf("Postfix expression:");      printf("%s",postfix);  }  int main() {      conversion();      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'3\_11\_infix\_to\_postfix\_conve\_stack.exe'  Enter infix expression: ((2\*3)+(4-2))/2  Postfix expression:23\*42-+2/  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program #3\_2** | **Date: 22/08/2022** |
| **Evaluate an expression using stack .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 3\_2 : EVALUATION OF EXPRESSION USING STACK  #include <stdio.h>  #include <string.h>  #include <math.h>  #define MAX 10  char stk[MAX];  int top = -1;  void push(char x) {      top++;      stk[top] = x;  }  char pop() {      char y = stk[top];      top--;      return y;  }  int precedence(char k) {      if (k == '^') {          return 3;      } else if (k == '\*' || k == '/') {          return 2;      } else if (k == '+' || k == '-') {          return 1;      } else {          return 0;      }  }  char conversion() {      char infix[MAX], postfix[MAX];      printf("Enter infix expression: ");      scanf("%s", infix);        int i = 0, j = 0;      char temp,k;      while (infix[i] != '\0') {          temp = infix[i];          switch (temp) {              case '(':                  push(temp);                  break;              case ')':                   k = pop();                  while (k != '(') {                      postfix[j] = k;                      j++;                      k = pop();                  }                  break;              case '^':              case '\*':              case '/':              case '+':              case '-':                  while (precedence(stk[top]) >= precedence(temp)) {                      postfix[j] = pop();                      j++;                  }                  push(temp);                  break;              default:                  postfix[j] = temp;                  j++;          }          i++;      }      while (top > -1) {          postfix[j] = pop();          j++;      }      postfix[j] = '\0';      printf("\nPostfix expression:");      printf("%s",postfix);      int operand1, operand2;      i = 0;      while (postfix[i] != '\0') {          if (isdigit(postfix[i])) {              push(postfix[i] - '0'); // Convert char digit to int and push to stack          } else {              operand2 = pop();              operand1 = pop();              switch (postfix[i]) {                  case '^':                      push(pow(operand1, operand2));                      break;                  case '\*':                      push(operand1 \* operand2);                      break;                  case '/':                      push(operand1 / operand2);                      break;                  case '+':                      push(operand1 + operand2);                      break;                  case '-':                      push(operand1 - operand2);                      break;              }          }          i++;      }      int evaluationResult = pop();      printf("Expression Evaluation Result: %d\n", evaluationResult);  }  int main() {      conversion();      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'3\_2\_evaluate\_exp\_stack.exe'  Enter infix expression: ((2\*3)+(4-2))/2  Postfix expression:23\*42-+2/  Expression Evaluation Result: 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 4\_1** | **Date: 24/08/2022** |
| **Define a structure for dates with dd/mm/yyyy. Provide functions for reading, displaying and comparing two dates are equal or not .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 4\_1 : DEFINE STRUCTURE FOR DATES(PROVIDE FUNCTIONS FOR READING,DISPLAYING AND COMPARING TWO DATES )  #include<stdio.h>  struct Date  {      int day, month, year;  };  void readDate(struct Date \*date) {      printf("\nEnter day of date(dd) : ");      scanf("%d",&(\*date).day);      printf("\nEnter month of date(mm) : ");      scanf("%d",&(\*date).month);      printf("\nEnter year of date(yyyy) : ");      scanf("%d",&(\*date).year);  }  void printDate(struct Date date) {      printf("%2d/%2d/%2d\n", date.day, date.month, date.year);  }  int isDateEqual(struct Date d1, struct Date d2 )  {      return (d1.day == d2.day && d1.month == d2.month && d1.year == d2.year);  }  int main() {      struct Date d1,d2;      printf("\nEnter details of First date (dd/mm/yyyy) \n");      readDate(&d1);      printf("\nEnter details of Second date (dd/mm/yyyy) \n");      readDate(&d2);      printf("\nFirst date : ");      printDate(d1);      printf("\nSecond date : ");      printDate(d2);      if(isDateEqual(d1,d2))          printf("\nEntered Dates are Same");      else          printf("\nEntered Dates are not same");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'4\_1\_struct\_for\_dates.exe'  Enter details of First date (dd/mm/yyyy)  Enter day of date(dd) : 10  Enter month of date(mm) : 12  Enter year of date(yyyy) : 2020  Enter details of Second date (dd/mm/yyyy)  Enter day of date(dd) : 10  Enter month of date(mm) : 12  Enter year of date(yyyy) : 2020  First date : 10/12/2020  Second date : 10/12/2020  Entered Dates are Same  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 4\_2** | **Date: 24/08/2022** |
| **Define a structure for employees with eno,ename, esal and dno. Read n employees information and provide functions for the following:**   1. **Searching an employee by no** 2. **Sorting the employees by**    * 1. **Name**      2. **Salary** 3. **Deleting an employee** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 4\_2 : DEFINE STRUCTURE FOR EMPLOYEES(search,sort and delete)  #include <stdio.h>  #include <string.h>  #define MAX 100  struct Employee {      int eno;      char ename[30];      double esal;      int dno;  };  void insertEmp(struct Employee employees[],int i,int n){      if(i==n){          printf("\nMemory is full.Cannot insert");          return ;      }      printf("\nEnter the details of the Employee:----");      printf("\nEmployee Number : ");      scanf("%d",&employees[i].eno);      printf("Employee Name : ");      scanf("%s",&employees[i].ename);      printf("Employee Salary : ");      scanf("%lf",&employees[i].esal);      printf("Department Number : ");      scanf("%d",&employees[i].dno);    }  void searchEmpByNo(struct Employee employees[],int n,int eno){      int i;      for(i=0;i < n;i++){          if(employees[i].eno==eno){              printf("-------Searched Employee FOUND--------\n");              printf("Employee Number : %d\nEmployee Name : %s\nEmployee Salary : %f\nDepartment Number : %d\n",employees[i].eno,employees[i].ename,employees[i].esal,employees[i].dno);              return;          }      }      printf("Searched Employee with empno %d NOT FOUND.\n",eno);  }  void sortEmpByName(struct Employee employees[],int n){      int i,j;      for(i=0;i < n - 1;i++){          for(j=0;j < n - i - 1;j++){              if(strcmp(employees[j].ename,employees[j+1].ename) > 0){                  struct Employee temp = employees[j];                  employees[j] = employees[j+1];                  employees[j+1]=temp;              }          }      }    }  void sortEmpBySal(struct Employee employees[],int n){      int i ,j;      for (i=0; i < n - 1;i++){          for(j=0;j < n - i -1;j++){              if(employees[j].esal > employees[j+1].esal){                  struct Employee temp = employees[j];                  employees[j] = employees[j+1];                  employees[j+1] = temp;                }          }      }  }  int deleteEmpByNum(struct Employee employees[],int n,int eno){      int i,j;      for(i=0 ; i < n ; i++){          if(employees[i].eno==eno){              for(j=i;j < n - 1;j++){                  employees[j] = employees[j+1];              }              n--;              printf("Employee with eno: %d deleted SUCCESSFULLY.\n",eno);              return n;          }      }      printf("\nEmployee with eno : %d NOT FOUND.\n ",eno);      return n;  }  void displayEmp(struct Employee employees[],int n){      int i;      printf("Employee Details : \n");      printf("ENO\tENAME\tESALARY\t\tDNO\n");      for(i=0;i < n ;i++){          printf("%d\t%s\t%f\t%d\n",employees[i].eno,employees[i].ename,employees[i].esal,employees[i].dno);        }  }  int main(){      struct Employee employees[MAX];      int n,i=0;        printf("\nEnter the number of employees (MAX 100) : ");      scanf("%d",&n);        int choice,eno;      do{          printf("\n\_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_\n");          printf("1.INSERT \n2.SEARCH(by eno) \n3.SORT(by name) \n4.SORT(by salary) \n5.DELETE(by eno) \n6.DISPLAY \n7.EXIT \n\_\_\_\_\_\_\_\nEnter Your Choice : ");          scanf("%d",&choice);            switch(choice) {              case 1:                  insertEmp(employees,i,n);                  i++;                  break;                case 2:                  printf("\nEnter the Employee number to be searched : ");                  scanf("%d",&eno);                  searchEmpByNo(employees,i,eno);                  break;                case 3:                  sortEmpByName(employees,i);                  printf("Employees sorted by name SUCCESSFULLY.\n");                  break;                case 4:                  sortEmpBySal(employees,i);                  printf("Employees sorted by salary SUCCESSFULLY.\n");                  break;                case 5:                  printf("\nEnter the Employee number of employee to be deleted : ");                  scanf("%d",&eno);                  i = deleteEmpByNum(employees,i,eno);                  break;                case 6:                  displayEmp(employees,i);                  break;                case 7:                  printf("\nExiting..........\n");                  break;                default:                  printf("INVALID CHOICE.PLEASE TRY AGAIN\n");            }      }while(choice != 7 );      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'4\_2\_struct\_for\_employee.exe'  Enter the number of employees (MAX 100) : 3  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 1  Enter the details of the Employee:----  Employee Number : 101  Employee Name : nihal  Employee Salary : 95000  Department Number : 7  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 1  Enter the details of the Employee:----  Employee Number : 102  Employee Name : majo  Employee Salary : 55000  Department Number : 4  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 1  Enter the details of the Employee:----  Employee Number : 103  Employee Name : hari  Employee Salary : 68000  Department Number : 8  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 1  Memory is full.Cannot insert  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 2  Enter the Employee number to be searched : 102  -------Searched Employee FOUND--------  Employee Number : 102  Employee Name : majo  Employee Salary : 55000.000000  Department Number : 4  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 6  Employee Details :  ENO ENAME ESALARY DNO  101 nihal 95000.000000 7  102 majo 55000.000000 4  103 hari 68000.000000 8  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 3  Employees sorted by name SUCCESSFULLY.  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 6  Employee Details :  ENO ENAME ESALARY DNO  103 hari 68000.000000 8  102 majo 55000.000000 4  101 nihal 95000.000000 7  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 4  Employees sorted by salary SUCCESSFULLY.  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 6  Employee Details :  ENO ENAME ESALARY DNO  102 majo 55000.000000 4  103 hari 68000.000000 8  101 nihal 95000.000000 7  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 5  Enter the Employee number of employee to be deleted : 101  Employee with eno: 101 deleted SUCCESSFULLY.  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 6  Employee Details :  ENO ENAME ESALARY DNO  102 majo 55000.000000 4  103 hari 68000.000000 8  \_\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_  1.INSERT  2.SEARCH(by eno)  3.SORT(by name)  4.SORT(by salary)  5.DELETE(by eno)  6.DISPLAY  7.EXIT  \_\_\_\_\_\_\_  Enter Your Choice : 7  Exiting..........  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 5\_1** | **Date: 26/08/2022** |
| **Read a polynomial and display it using array.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 5\_1 : READ A POLYNOMIAL AND DISPLAY IT USING ARRAY  #include <stdio.h>  int read();  void sort();  void disp();  int main(){      int n=0,coef[20],expon[20];      n=read(coef,expon,n);      disp(coef,expon,n);      return 0;  }  int read(int coef[],int expon[],int n){      int i;      printf("\nEnter the number of terms in the expression : ");      scanf("%d",&n);      printf("\n\_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_\n");      for(i=0;i<n;i++){          printf("\nEnter details of term %d :-\nCoefficient : ",i+1);          scanf("%d",&coef[i]);          printf("\nExponent : ");          scanf("%d",&expon[i]);      }      return n;    }  void sort(int coef[],int expon[],int n){      int i,j,t1,t2;      for(i=0;i<n;i++){          for(j=0;j<n-i-1;j++){              if(expon[j]<expon[j+1]){                  t1=expon[j];                  expon[j]=expon[j+1];                  expon[j+1]=t1;                    t2=coef[j];                  coef[j]=coef[j+1];                  coef[j+1]=t2;              }          }      }  }  void disp(int coef[],int expon[],int n){      int i;      sort(coef,expon,n);      printf("\nPolynomial is : ");      for(i=0;i<n;i++){          if(expon[i]==0){              if(coef[i]<0){                  printf(" - %d",-coef[i]);              }              else if(coef[i]>0){                  if(i==0){                      printf("%d",coef[i]);                  }else{                      printf(" + %d",coef[i]);                  }              }          }          else if(expon[i]>0){              if(coef[i]<0){                  printf(" - %dx^%d",-coef[i],expon[i]);              }              else if(coef[i]>0){                  if(i==0){                      printf("%dx^%d",coef[i],expon[i]);                  }else{                      printf(" + %dx^%d",coef[i],expon[i]);                  }              }          }          else{              if(coef[i]<0){                  printf(" - %dx^%d",-coef[i],-expon[i]);              }else if(coef[i]>0){                  if(i==0){                  printf("%d/x^%d",coef[i],-expon[i]);                  }              }else{                  printf(" + %d/x^%d",coef[i],-expon[i]);              }          }      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'5\_1\_polynomial\_read\_display.exe'  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : -62  Exponent : 2  Enter details of term 2 :-  Coefficient : 20  Exponent : 1  Enter details of term 3 :-  Coefficient : 5  Exponent : 0  Polynomial is : - 62x^2 + 20x^1 + 5  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 5\_2** | **Date: 27/08/2022** |
| **Add two polynomials using the array itself .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 5\_2 : ADD TWO POLYNOMIALS USING ARRAY  #include <stdio.h>  int n1,n2,n3,coef1[20],expon1[20],coef2[20],expon2[20],coef3[20],expon3[20];  int read();  void sort();  void disp();  void add();  int main(){      int n=0;      n1=read(coef1,expon1,n);      sort(coef1,expon1,n1);      disp(coef1,expon1,n1);        n2=read(coef2,expon2,n);      sort(coef2,expon2,n2);      disp(coef2,expon2,n2);        printf("\n\nNow , Adding two polynomials :  ");      add();      sort(coef3,expon3,n3);      disp(coef3,expon3,n3);      return 0;  }  int read(int coef[],int expon[],int n){      int i;      printf("\nEnter the number of terms in the expression : ");      scanf("%d",&n);      printf("\n\_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_\n");      for(i=0;i<n;i++){          printf("\nEnter details of term %d :-\nCoefficient : ",i+1);          scanf("%d",&coef[i]);          printf("\nExponent : ");          scanf("%d",&expon[i]);      }      return n;    }  void sort(int coef[],int expon[],int n){      int i,j,t1,t2;      for(i=0;i<n;i++){          for(j=0;j<n-i-1;j++){              if(expon[j]<expon[j+1]){                  t1=expon[j];                  expon[j]=expon[j+1];                  expon[j+1]=t1;                    t2=coef[j];                  coef[j]=coef[j+1];                  coef[j+1]=t2;              }          }      }  }  void disp(int coef[],int expon[],int n){      int i;      sort(coef,expon,n);      printf("\nPolynomial is : ");      for(i=0;i<n;i++){          if(expon[i]==0){              if(coef[i]<0){                  printf(" - %d",-coef[i]);              }              else if(coef[i]>0){                  if(i==0){                      printf("%d",coef[i]);                  }else{                      printf(" + %d",coef[i]);                  }              }          }          else if(expon[i]>0){              if(coef[i]<0){                  printf(" - %dx^%d",-coef[i],expon[i]);              }              else if(coef[i]>0){                  if(i==0){                      printf("%dx^%d",coef[i],expon[i]);                  }else{                      printf(" + %dx^%d",coef[i],expon[i]);                  }              }          }          else{              if(coef[i]<0){                  printf(" - %dx^%d",-coef[i],-expon[i]);              }else if(coef[i]>0){                  if(i==0){                  printf("%d/x^%d",coef[i],-expon[i]);                  }              }else{                  printf(" + %d/x^%d",coef[i],-expon[i]);              }          }      }  }  void add(){      int i=0,j=0,k=0;      while(i<n1 && j<n2){          if(expon1[i] == expon2[j])          {              coef3[k] = coef1[i] + coef2[j];              expon3[k] = expon1[i];              i++;              j++;              k++;          }          else if(expon1[i] > expon2[j]){              coef3[k] = coef1[i];              expon3[k] = expon1[i];              i++;              k++;          }          else          {              coef3[k] = coef2[j];              expon3[k] = expon2[j];              j++;              k++;          }      }      while(i<n1){          coef3[k] = coef1[i];          expon3[k] = expon1[i];          i++;          k++;      }      while(j<n2){          coef3[k] = coef2[j];          expon3[k] = expon2[j];          j++;          k++;      }      n3=k;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'5\_2\_polynomial\_add.exe'  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : -62  Exponent : 2  Enter details of term 2 :-  Coefficient : 20  Exponent : 1  Enter details of term 3 :-  Coefficient : 5  Exponent : 0  Polynomial is : - 62x^2 + 20x^1 + 5  Enter details of other polynomial to add ---->  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : 40  Exponent : 2  Enter details of term 2 :-  Coefficient : 25  Exponent : 1  Enter details of term 3 :-  Coefficient : 15  Exponent : 0  Polynomial is : 40x^2 + 25x^1 + 15  Now , Adding two polynomials :  Polynomial is : - 22x^2 + 45x^1 + 20  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 6\_1** | **Date: 28/08/2022** |
| **Read a polynomial and display it using structure array** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 6\_1 : Read a polynomial and display it using structure array  #include<stdio.h>  struct poly{      int coef;      int expon;  };  int read();  void sort();  void disp();  int main(){      struct poly p[10];      int n;      n=read(p);      disp(p,n);      return 0;  }  int read(struct poly p[]){      int i,n;      printf("\nEnter the number of terms in the expression : ");      scanf("%d",&n);      printf("\n\_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_\n");      for(i=0;i<n;i++){          printf("\nEnter details of term %d :-\nCoefficient : ",i+1);          scanf("%d",&p[i].coef);          printf("\nExponent : ");          scanf("%d",&p[i].expon);      }      return n;  }  void sort(struct poly p[],int n){      int i,j;      struct poly temp;      for(i=0;i<n;i++){          for(j=0;j<n-i-1;j++){              if(p[j].expon<p[j+1].expon){                  temp=p[j];                  p[j]=p[j+1];                  p[j+1]=temp;              }          }      }  }  void disp(struct poly p[],int n){      int i;      sort(p,n);      printf("\nPolynomial is : ");      for(i=0;i<n;i++){          if(p[i].expon==0){              if(p[i].coef<0){                  printf(" - %d",-p[i].coef);              }              else if(p[i].coef>0){                  if(i==0){                      printf("%d",p[i].coef);                  }else{                      printf(" + %d",p[i].coef);                  }              }          }          else if(p[i].expon>0){              if(p[i].coef<0){                  printf(" - %dx^%d",-p[i].coef,p[i].expon);              }              else if(p[i].coef>0){                  if(i==0){                      printf("%dx^%d",p[i].coef,p[i].expon);                  }else{                      printf(" + %dx^%d",p[i].coef,p[i].expon);                  }              }          }          else{              if(p[i].coef<0){                  printf(" - %dx^%d",-p[i].coef,-p[i].expon);              }else if(p[i].coef>0){                  if(i==0){                  printf("%d/x^%d",p[i].coef,-p[i].expon);                  }              }else{                  printf(" + %d/x^%d",p[i].coef,-p[i].expon);              }          }      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'6\_1\_read\_disp\_pol\_struct.exe'  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : -22  Exponent : 2  Enter details of term 2 :-  Coefficient : 10  Exponent : 1  Enter details of term 3 :-  Coefficient : 5  Exponent : 0  Polynomial is : - 22x^2 + 10x^1 + 5  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 6\_2** | **Date: 28/08/2022** |
| **Add two polynomials using structure array.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 6\_2 :Add two polynomials using structure  #include<stdio.h>  struct poly{      int coef;      int expon;  };  int read();  void sort();  void disp();  int add();  int main(){      struct poly p1[10],p2[10],p3[10];      int n1,n2,n3;      printf("\nEnter details of FIRST polynomial to add ---->\n");      n1=read(p1);      disp(p1,n1);      printf("\n\nEnter details of SECOND polynomial to add ---->\n");      n2=read(p2);      disp(p2,n2);      printf("\n\nAdding the given two polynomials............");      n3=add(p1,p2,p3,n1,n2);      disp(p3,n3);      return 0;  }  int read(struct poly p[]){      int i,n;      printf("\nEnter the number of terms in the expression : ");      scanf("%d",&n);      printf("\n\_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_\n");      for(i=0;i<n;i++){          printf("\nEnter details of term %d :-\nCoefficient : ",i+1);          scanf("%d",&p[i].coef);          printf("\nExponent : ");          scanf("%d",&p[i].expon);      }      return n;  }  void sort(struct poly p[],int n){      int i,j;      struct poly temp;      for(i=0;i<n;i++){          for(j=0;j<n-i-1;j++){              if(p[j].expon<p[j+1].expon){                  temp=p[j];                  p[j]=p[j+1];                  p[j+1]=temp;              }          }      }  }  void disp(struct poly p[],int n){      int i;      sort(p,n);      printf("\nPolynomial is : ");      for(i=0;i<n;i++){          if(p[i].expon==0){              if(p[i].coef<0){                  printf(" - %d",-p[i].coef);              }              else if(p[i].coef>0){                  if(i==0){                      printf("%d",p[i].coef);                  }else{                      printf(" + %d",p[i].coef);                  }              }          }          else if(p[i].expon>0){              if(p[i].coef<0){                  printf(" - %dx^%d",-p[i].coef,p[i].expon);              }              else if(p[i].coef>0){                  if(i==0){                      printf("%dx^%d",p[i].coef,p[i].expon);                  }else{                      printf(" + %dx^%d",p[i].coef,p[i].expon);                  }              }          }          else{              if(p[i].coef<0){                  printf(" - %dx^%d",-p[i].coef,-p[i].expon);              }else if(p[i].coef>0){                  if(i==0){                  printf("%d/x^%d",p[i].coef,-p[i].expon);                  }              }else{                  printf(" + %d/x^%d",p[i].coef,-p[i].expon);              }          }      }  }  int add(struct poly p1[10], struct poly p2[10], struct poly p3[10], int n1, int n2)  {      int i=0, j=0, k=0;      while (i < n1 && j < n2)      {          if (p1[i].expon== p2[j].expon)          {              p3[k].coef = p1[i].coef + p2[j].coef;              p3[k].expon = p1[i].expon;              i++;              j++;              k++;          }          else if (p1[i].expon > p2[j].expon)          {              p3[k].coef = p1[i].coef;              p3[k].expon = p1[i].expon;              i++;              k++;          }          else          {              p3[k].coef = p2[j].coef;              p3[k].expon = p2[j].expon;              j++;              k++;          }      }      while (i < n1)      {          p3[k].coef = p1[i].coef;          p3[k].expon = p1[i].expon;          i++;          k++;      }      while (j < n2)      {          p3[k].coef = p2[j].coef;          p3[k].expon = p2[j].expon;          j++;          k++;      }      return (k);  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'6\_2\_add\_pol\_struct.exe'  Enter details of FIRST polynomial to add ---->  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : 60  Exponent : 2  Enter details of term 2 :-  Coefficient : -20  Exponent : 1  Enter details of term 3 :-  Coefficient : -5  Exponent : 0  Polynomial is : 60x^2 - 20x^1 - 5  Enter details of SECOND polynomial to add ---->  Enter the number of terms in the expression : 2  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : 25  Exponent : 1  Enter details of term 2 :-  Coefficient : 18  Exponent : 0  Polynomial is : 25x^1 + 18  Adding the given two polynomials............  Polynomial is : 60x^2 + 5x^1 + 13  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 6\_3** | **Date: 29/08/2022** |
| **Subtract two polynomials .** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 6\_3 : Subtract two polynomials using structure  #include<stdio.h>  struct poly{      int coef;      int expon;  };  int read();  void sort();  void disp();  void minusof();  int subtract();  int main(){      struct poly p1[10],p2[10],p3[10];      int n1,n2,n3;      printf("\nEnter details of FIRST polynomial to subtract ---->\n");      n1=read(p1);      disp(p1,n1);      printf("\n\nEnter details of SECOND polynomial to subtract ---->\n");      n2=read(p2);      disp(p2,n2);      printf("\n\nSubtracting the first polynomial from second............\n");      n3=subtract(p1,p2,p3,n1,n2);      disp(p3,n3);      return 0;  }  int read(struct poly p[]){      int i,n;      printf("\nEnter the number of terms in the expression : ");      scanf("%d",&n);      printf("\n\_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_\n");      for(i=0;i<n;i++){          printf("\nEnter details of term %d :-\nCoefficient : ",i+1);          scanf("%d",&p[i].coef);          printf("\nExponent : ");          scanf("%d",&p[i].expon);      }      return n;  }  void sort(struct poly p[],int n){      int i,j;      struct poly temp;      for(i=0;i<n;i++){          for(j=0;j<n-i-1;j++){              if(p[j].expon<p[j+1].expon){                  temp=p[j];                  p[j]=p[j+1];                  p[j+1]=temp;              }          }      }  }  void disp(struct poly p[],int n){      int i;      sort(p,n);      printf("\nPolynomial is : ");      for(i=0;i<n;i++){          if(p[i].expon==0){              if(p[i].coef<0){                  printf(" - %d",-p[i].coef);              }              else if(p[i].coef>0){                  if(i==0){                      printf("%d",p[i].coef);                  }else{                      printf(" + %d",p[i].coef);                  }              }          }          else if(p[i].expon>0){              if(p[i].coef<0){                  printf(" - %dx^%d",-p[i].coef,p[i].expon);              }              else if(p[i].coef>0){                  if(i==0){                      printf("%dx^%d",p[i].coef,p[i].expon);                  }else{                      printf(" + %dx^%d",p[i].coef,p[i].expon);                  }              }          }          else{              if(p[i].coef<0){                  printf(" - %dx^%d",-p[i].coef,-p[i].expon);              }else if(p[i].coef>0){                  if(i==0){                  printf("%d/x^%d",p[i].coef,-p[i].expon);                  }              }else{                  printf(" + %d/x^%d",p[i].coef,-p[i].expon);              }          }      }  }  void minusof(struct poly p[10],int n){      int i;      for(i=0;i<n;i++){          p[i].coef=-p[i].coef;      }  }  int subtract(struct poly p1[10], struct poly p2[10], struct poly p3[10], int n1, int n2)  {      minusof(p2,n2);      int i=0, j=0, k=0;      while (i < n1 && j < n2)      {          if (p1[i].expon== p2[j].expon)          {              p3[k].coef = p1[i].coef + p2[j].coef;              p3[k].expon = p1[i].expon;              i++;              j++;              k++;          }          else if (p1[i].expon > p2[j].expon)          {              p3[k].coef = p1[i].coef;              p3[k].expon = p1[i].expon;              i++;              k++;          }          else          {              p3[k].coef = p2[j].coef;              p3[k].expon = p2[j].expon;              j++;              k++;          }      }      while (i < n1)      {          p3[k].coef = p1[i].coef;          p3[k].expon = p1[i].expon;          i++;          k++;      }      while (j < n2)      {          p3[k].coef = p2[j].coef;          p3[k].expon = p2[j].expon;          j++;          k++;      }      return (k);  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'6\_3\_subtract\_pol\_struct.exe'  Enter details of FIRST polynomial to subtract ---->  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : 60  Exponent : 2  Enter details of term 2 :-  Coefficient : 30  Exponent : 1  Enter details of term 3 :-  Coefficient : 5  Exponent : 0  Polynomial is : 60x^2 + 30x^1 + 5  Enter details of SECOND polynomial to subtract ---->  Enter the number of terms in the expression : 3  \_\_\_\_\_\_\_\_Enter the polynomial expression\_\_\_\_\_\_\_\_\_\_  Enter details of term 1 :-  Coefficient : 10  Exponent : 2  Enter details of term 2 :-  Coefficient : 10  Exponent : 1  Enter details of term 3 :-  Coefficient : 2  Exponent : 0  Polynomial is : 10x^2 + 10x^1 + 2  Subtracting the first polynomial from second............  Polynomial is : 50x^2 + 20x^1 + 3  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 6\_4** | **Date: 29/08/2022** |
| **Multiply two polynomials.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 6\_4 : Multiply two polynomials  #include<stdio.h>  struct poly  {      int coeff;      int expo;  };  int read(struct poly p[])  {      int i,n;      printf("\nEnter the number of terms in the expression: ");      scanf("%d",&n);      printf("\nEnter a polynomial expression:\n------------------------");      for(i=0;i<n;i++)      {          printf("\nEnter term %d:\nCoefficient: ",i+1);          scanf("%d",&p[i].coeff);          printf("\nExponent: ");          scanf("%d",&p[i].expo);      }      return n;  }  void sort(struct poly p[], int n)  {      int i,j;      struct poly temp;      for(i=0;i<n;i++)      {          for(j=0;j<n-i-1;j++)          {              if(p[j].expo<p[j+1].expo)              {                  temp=p[j];                  p[j]=p[j+1];                  p[j+1]=temp;              }          }      }  }  void disp(struct poly p[], int n)  {      int i;      sort(p,n);      printf("\nPolynomial is: ");      for(i=0;i<n;i++)      {          if(p[i].expo==0)          {              if(p[i].coeff<0)                  printf(" - %d",-p[i].coeff);              else if(p[i].coeff>0)              {                  if(i==0)                      printf("%d",p[i].coeff);                  else                      printf(" + %d",p[i].coeff);              }          }          else if(p[i].expo>0)          {              if(p[i].coeff<0)                  printf(" - %dx^%d",-p[i].coeff,p[i].expo);              else if(p[i].coeff>0)              {                  if(i==0)                      printf("%dx^%d",p[i].coeff,p[i].expo);                  else                      printf(" + %dx^%d",p[i].coeff,p[i].expo);              }          }          else          {              if(p[i].coeff<0)                  printf(" - %d/x^%d",-p[i].coeff,-p[i].expo);              else if(p[i].coeff>0)              {                  if(i==0)                      printf("%d/x^%d",p[i].coeff,-p[i].expo);                  else                      printf(" + %d/x^%d",p[i].coeff,-p[i].expo);              }          }      }  }  void multiply(struct poly p1[], struct poly p2[], struct poly p3[], int n1, int n2)  {      int n3,i,j,k=0;      for(i=0;i<n1;i++)      {          for(j=0;j<n2;j++)          {              p3[k].expo=p1[i].expo+p2[j].expo;              p3[k].coeff=p1[i].coeff\*p2[j].coeff;              k++;          }      }      n3=k;      i=0;      sort(p3,n3);      while(i<n3-1)      {          if(p3[i].expo==p3[i+1].expo)          {              p3[i].coeff=p3[i].coeff+p3[i+1].coeff;              j=i+1;              while(j<n3-1)              {                  p3[j].expo=p3[j+1].expo;                  p3[j].coeff=p3[j+1].coeff;                  j++;              }              n3--;          }          else          {              i++;          }      }      disp(p3,n3);  }  int main()  {      struct poly p1[10], p2[10], p3[20];      int n1, n2;      printf("\nEnter the details of First Polynomial--->\n");      n1=read(p1);      disp(p1,n1);      printf("\n\nEnter the details of  Second Polynomial--->\n");      n2=read(p2);      disp(p2,n2);      printf("\nProduct\n-------");      multiply(p1,p2,p3,n1,n2);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'6\_4\_multply\_pol\_struct.exe'  Enter the details of First Polynomial--->  Enter the number of terms in the expression: 3  Enter a polynomial expression:  ------------------------  Enter term 1:  Coefficient: 3  Exponent: 2  Enter term 2:  Coefficient: 2  Exponent: 1  Enter term 3:  Coefficient: 1  Exponent: 0  Polynomial is: 3x^2 + 2x^1 + 1  Enter the details of Second Polynomial--->  Enter the number of terms in the expression: 3  Enter a polynomial expression:  ------------------------  Enter term 1:  Coefficient: 1  Exponent: 2  Enter term 2:  Coefficient: 1  Exponent: 1  Enter term 3:  Coefficient: 2  Exponent: 0  Polynomial is: 1x^2 + 1x^1 + 2  Product  -------  Polynomial is: 3x^4 + 5x^3 + 9x^2 + 5x^1 + 2  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 7\_1** | **Date: 01/09/2022** |
| **Implement: a) malloc , b) calloc and c) free functions.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 7\_1 : Implement a) malloc , b) calloc and c) free functions  #include<stdio.h>  #include<malloc.h>  int main(){      int n,i,\*p1,\*p2;      printf("\nDynamic memory allocation.\nEnter value for n(how many memory locations?): ");      scanf("%d",&n);      p1=(int \*)malloc(sizeof(int)\*n);//Allocating dynamic memory using malloc()      p2=(int \*)calloc(n,sizeof(int));//Allocating dynamic memory using calloc()        if(p1!=NULL){          printf("\nmalloc-Dynamic memory allocation successfull.\nEnter %d values : ",n);          for(i=0;i<n;i++){              printf("\nEnter p1[%d]: ",i);              scanf("%d",p1+i);          }          printf("\nEntered values are : \t");          for(i=0;i<n;i++){              printf("%d\t",\*(p1+i));          }      }      else{          printf("\nmalloc-Dynamic memory allocation Failed");      }      if(p2!=NULL){          printf("\n\ncalloc-Dynamic memory allocation successfull.\nEnter %d values : ",n);          for(i=0;i<n;i++){              printf("\nEnter p1[%d]: ",i);              scanf("%d",p1+i);          }          printf("\nEntered values are : \t");          for(i=0;i<n;i++){              printf("%d\t",\*(p1+i));          }      }      else{          printf("\ncalloc-Dynamic memory allocation Failed");      }      /\*The memory allocated will not be released automatically after using memory.      The space remains there and cannot be used\*/      free(p1);//Deallocating or Releasing memory using free()      free(p2);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'7\_1\_impl\_malloc\_calloc\_free.exe'  Dynamic memory allocation.  Enter value for n(how many memory locations?): 5  malloc-Dynamic memory allocation successfull.  Enter 5 values :  Enter p1[0]: 10  Enter p1[1]: 20  Enter p1[2]: 30  Enter p1[3]: 40  Enter p1[4]: 50  Entered values are : 10 20 30 40 50  calloc-Dynamic memory allocation successfull.  Enter 5 values :  Enter p1[0]: 60  Enter p1[1]: 70  Enter p1[2]: 80  Enter p1[3]: 90  Enter p1[4]: 92  Entered values are : 60 70 80 90 92  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 7\_2** | **Date: 01/09/2022** |
| **Use malloc to read n integers and find the mean.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 7\_2 : Use malloc to read n integers and find the mean.  #include<stdio.h>  #include<malloc.h>  int main(){      int n,x=0,i,\*p;      float mean;      printf("\nEnter how much integers to enter(value for n): ");      scanf("%d",&n);      p=(int \*)malloc(sizeof(int)\*n);//Allocating dynamic memory using malloc()      printf("\nEnter values for integers -> ");      for(i=0;i<n;i++){              printf("\nEnter value of integer %d: ",i+1);              scanf("%d",p+i);              x=x+\*(p+i);          }      printf("\nEntered integer values are : ");      for(i=0;i<n;i++){              printf("\t%d",\*(p+i));          }      mean=x/n;      printf("\nMean of given integer values = %f",mean);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'7\_2\_malloc\_read\_mean.exe'  Enter how much integers to enter(value for n): 5  Enter values for integers ->  Enter value of integer 1: 10  Enter value of integer 2: 20  Enter value of integer 3: 30  Enter value of integer 4: 40  Enter value of integer 5: 50  Entered integer values are : 10 20 30 40 50  Mean of given integer values = 30.000000  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 7\_3** | **Date: 02/09/2022** |
| **Use calloc to read n numbers and find the mode.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 7\_3 :Use calloc to read n numbers and find the mode.  #include<stdio.h>  #include<malloc.h>  int i,j;  int mode\_fn(int \*p,int n){      int a[n],k,count,max\_count=0;      for(i=0;i<n;i++){          count=0;          for(j=i;j<n;j++){              if(\*(p+i)==\*(p+j))                  count++;          }          if(count>=max\_count){              if(count>max\_count)                  k=0;              a[k]=\*(p+i);              max\_count=count;              k++;          }      }      //when there is only one mode      if(k==1){          printf("\nMode of given data = %d",a[0]);      }      //when there is more than one modes      else      {          printf("\nmodes of given data are : ");          for(i=0;i<k;i++)              printf("\t%d",a[i]);      }    }  int main(){      int n,\*p;      printf("\nEnter n(how much number to read) : ");      scanf("%d",&n);      p=(int \*)calloc(n,sizeof(int));//Allocating dynamic memory using malloc()      printf("\nEnter %d numbers -> ",n);      for(i=0;i<n;i++){          printf("\nEnter number %d : ",i+1);          scanf("%d",p+i);      }      printf("\nEntered values are : ");      //displaying values      for(i=0;i<n;i++){          printf("\t%d",\*(p+i));      }      mode\_fn(p,n);//deallocating memory      free(p);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'7\_3\_calloc\_read\_mode.exe'  Enter n(how much number to read) : 5  Enter 5 numbers ->  Enter number 1 : 10  Enter number 2 : 20  Enter number 3 : 30  Enter number 4 : 40  Enter number 5 : 20  Entered values are : 10 20 30 40 20  Mode of given data = 20  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 7\_4** | **Date: 02/09/2022** |
| **Declare a structure for Books having author\_name and book\_name. Create an array of books using a pointer variable. Provide functions for reading n books and displaying the same using pointers.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 7\_4 :Declare a structure for Books having author\_name and book\_name. Create an array of books using a pointer variable. Provide functions for reading n books and displaying the same using pointers.  #include<stdio.h>  #include<malloc.h>  struct books{      char author\_name[30];      char book\_name[50];  };  void read();  void disp();  int main(){      int n;      struct books \*p;      printf("\nEnter the number of books : ");      scanf("%d",&n);      p=(struct books \*)calloc(n,sizeof(struct books));      read(p,n);      disp(p,n);      return 0;  }  void read(struct books \*p,int n){      int i;      printf("\n----------Enter the details of books------------");      for(i=0;i<n;i++){          printf("\n\n\nBook %d ------------>",i+1);          printf("\nEnter name of book : ");          scanf(" %[^\n]",(p + i)->book\_name);//[^\n] : Reads strings with whitespaces          printf("\nEnter the name of author : ");          scanf(" %[^\n]",(p + i)->author\_name);        }  }  void disp(struct books \*p,int n){      int i;      printf("\n---------------BOOK DETAILS----------------");      for(i=0;i<n;i++){          printf("\n\n\nBook %d ------------>",i+1);          printf("\nName of Book : %s\n",(p+i)->book\_name);          printf("\nName of author : %s\n",(p+i)->author\_name);      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'7\_4\_struct\_books\_pointers.exe'  Enter the number of books : 3  ----------Enter the details of books------------  Book 1 ------------>  Enter name of book : Datastructure using C  Enter the name of author : E Balaguruswamy  Book 2 ------------>  Enter name of book : DBMS  Enter the name of author : A Madhavi  Book 3 ------------>  Enter name of book : DCCN  Enter the name of author : Achyuth S Godbole  ---------------BOOK DETAILS----------------  Book 1 ------------>  Name of Book : Datastructure using C  Name of author : E Balaguruswamy  Book 2 ------------>  Name of Book : DBMS  Name of author : A Madhavi  Book 3 ------------>  Name of Book : DCCN  Name of author : Achyuth S Godbole  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 7\_5** | **Date: 03/09/2022** |
| **Use realloc to implement varchar for any length.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 7\_5 :Use realloc to implement varchar for any length.  #include<stdio.h>  #include<malloc.h>  int main(){      int i=0;      char \*ptr,\*ptr\_new,ch;      printf("\nEnter a string : ");      ptr=(char \*)malloc(sizeof(char)\*2);      do{          scanf("%c",&ch);          \*(ptr+i)=ch;          i++;          ptr\_new =(char \*)realloc(ptr,sizeof(char)\*(i+2));      }while(ch !='\n');        \*(ptr+i)='\0';//adding \0 to make it as string      printf("\nThe entered string is : %s",ptr\_new);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'7\_5\_realloc\_to\_imple\_varchar.exe'  Enter a string : anshad  The entered string is : anshad  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 8\_1** | **Date: 05/09/2022** |
| **Implement Queue using array.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 8\_1 :Implement Queue using array.  #include<stdio.h>  #define MAX\_V 100  int q[MAX\_V];  int f=-1,r=-1,i;  void enqueue();  void dequeue();  void disp();  int menu();  int main(){      int ch,n;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to enqueue(n) : ");                  scanf("%d",&n);                  for(i=0;i<n;i++){                      enqueue();                  }                  break;              case 2:                  dequeue();                  break;              case 3:                  disp();                  break;              case 4:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nQUEUE OPERATIONS->\n1.ENQUEUE.\n2.DEQUEUE.\n3.DISPLAY.\n4.EXIT");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void enqueue(){      if(r+1==MAX\_V){          printf("\nQUEUE IS FULL.CANNOT PERFORM ENQUEUE\n");      }      else{          if(f==-1){              f=0;          }          printf("\nEnter the value to ENQUEUE : ");          scanf("%d",&q[++r]);//incrementing value of r each time when enqueue happens      }  }  void dequeue(){      if(f==-1){          printf("\nQUEUE IS EMPTY.CANNOT PERFORM DEQUEUE");        }else{          printf("\nDEQUEUED element : %d ",q[f]);          //When there is only one element          if(f==r){              f=r=-1;          }          //when there is more than one element in queue          else{              for(i=0;i<r;i++){                  q[i]=q[i+1];              }              r--;//decrement value of r when dequeue happens          }      }  }  void disp(){      if(f==-1){          printf("\nQUEUE IS EMPTY.");      }      else{              printf("\nDisplaying elements in QUEUE : ");              for(i=0;i<=r;i++){                  printf("\t%d",q[i]);              }      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'8\_1\_queue\_array.exe'  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 2  QUEUE IS EMPTY.CANNOT PERFORM DEQUEUE  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 1  Enter how many elements to enqueue(n) : 5  Enter the value to ENQUEUE : 10  Enter the value to ENQUEUE : 20  Enter the value to ENQUEUE : 30  Enter the value to ENQUEUE : 40  Enter the value to ENQUEUE : 50  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 3  Displaying elements in QUEUE : 10 20 30 40 50  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 2  DEQUEUED element : 10  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 2  DEQUEUED element : 20  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 3  Displaying elements in QUEUE : 30 40 50  QUEUE OPERATIONS->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 8\_2** | **Date: 07/09/2022** |
| **Implement priority queue.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 8\_2 :Implement priority queue  #include <stdio.h>  void heaptree(int a[], int n, int e)  {      int i, t;      i = n;      a[i] = e;      while (i != 0)      {          if (a[i] < a[(i - 1) / 2])          {              break;          }          else          {              t = a[i];              a[i] = a[(i - 1) / 2];              a[(i - 1) / 2] = t;          }          i = (i - 1) / 2;      }  }  void disparr(int a[], int n)  {      int i;      for (i = 0; i <= n; i++)      {          printf("%d\t", a[i]);      }  }  void heapsort(int a[], int n)  {      int i, j, t, lc, rc;      j = n;      while (j > 0)      {          i = 0;          t = a[i];          a[i] = a[j];          a[j] = t;          j--;          while (i <= j)          {              lc = (2 \* i) + 1;              rc = (2 \* i) + 2;              if (lc <= j && rc <= j)              {                  if ((a[i] > a[lc]) && (a[i] > a[rc]))                  {                      break;                  }                  else if (a[lc] > a[rc])                  {                      t = a[i];                      a[i] = a[lc];                      a[lc] = t;                      i = lc;                  }                  else                  {                      t = a[i];                      a[i] = a[rc];                      a[rc] = t;                      i = rc;                  }              }              else if (lc <= j && a[lc] > a[i])              {                  t = a[i];                  a[i] = a[lc];                  a[lc] = t;                  i = lc;              }              else              {                  break;              }              disparr(a, 4);              printf("\n");          }      }  }  int main()  {      int a[5];      heaptree(a, 0, 4);      heaptree(a, 1, 8);      heaptree(a, 2, 15);      heaptree(a, 3, 5);      heaptree(a, 4, 1);      disparr(a, 4);      printf("\n");      heapsort(a, 4);      disparr(a, 4);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'8\_2\_priority\_queue.exe'  15 5 8 4 1  8 5 1 4 15  5 4 1 8 15  4 1 5 8 15  1 4 5 8 15  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 9\_1** | **Date: 09/09/2022** |
| **Demonstrate a linked list creation and display.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 9\_1 :Demonstrate a linked list creation and display.  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  struct node \*head=NULL;  void insert(int e){      struct node \*t; //creating another pointer for traversing to last element      if(head==NULL){          head=(struct node \*)malloc(sizeof(struct node));  //1.allocating memory to head          head->data=e; //2.copying element to head->data          head->next=NULL; //3.making next of head as NULL      }      else{          t=head;          while(t->next != NULL) //1.Traversing upto the last element              t=t->next;          t->next=(struct node \*)malloc(sizeof(struct node)); //2.Allocating memory to the next of last node          t->next->data=e; //3.copying element to newly created node's data part          t->next->next=NULL;  //4.making next of newly created node as NULL      }  }  void disp(){      struct node \*t;      if(head==NULL)          printf("\nLinked list is empty.\n");      else{          t=head;          printf("\nDisplaying Created Linked List : \n");          while(t!=NULL){              printf("\t%d",t->data);              t=t->next;          }      }printf("\n");  }  int menu()  {      int ch;      printf("LINKED LIST OPERATIONS->\n1.INSERT.\n2.DISPLAY.\n3.EXIT");      printf("\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  int main(){      int ch,n,e,i;      for(ch=menu();ch!=3;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert(e);                  }                  break;              case 2:                  disp();                  break;              case 3:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'9\_1\_linkedlist\_creation\_display.exe'  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.EXIT  Enter any choice : 1  Enter how many elements to insert(n) : 5  Enter elements :  15  23  20  75  10  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.EXIT  Enter any choice : 2  Displaying Created Linked List :  15 23 20 75 10  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.EXIT  Enter any choice : 3  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 9\_2** | **Date: 10/09/2022** |
| **Write a program with functions to insert a new node**   * 1. **at the beginning of a Singly Linked List.**   2. **At the end of the linked list**   3. **after a specified element in a linked list.** | |

**Source Code:**

|  |
| --- |
| /\*PROGRAM 9\_2 :Write a program with functions  to insert a new node  a.at the beginning of a Singly Linked List.  b.At the end of the linked list  c.after a specified element in a linked list.  \*/  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  struct node \*head=NULL;  void insert\_at\_begin();  void insert\_at\_end();  void insert\_after\_el();  void disp();  int menu();  int main(){      int ch,n,e,i;      for(ch=menu();ch!=5;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter element to insert at the beginning of linked list : ");                  scanf("%d",&e);                  insert\_at\_begin(e);                  break;              case 2:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to insert at the end of linked list : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert\_at\_end(e);                  }                  break;              case 3:                  printf("\nEnter the element to search : ");                  scanf("%d",&e);                  insert\_after\_el(e);                  break;              case 4:                  disp();                  break;              case 5:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nLINKED LIST OPERATIONS->\n1.INSERT AT BEGINNING.\n2.INSERT AT END.\n3.INSERT AFTER A SPECIFIED ELEMENT.\n4.DISPLAY.\n5.EXIT");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void insert\_at\_begin(int e){      struct node \*t;      t=(struct node \*)malloc(sizeof(struct node));      t->data=e;      t->next=head;      head=t;  }  void insert\_at\_end(int e){      struct node \*t; //creating another pointer for traversing to last element      if(head==NULL){          head=(struct node \*)malloc(sizeof(struct node));  //1.allocating memory to head          head->data=e; //2.copying element to head->data          head->next=NULL; //3.making next of head as NULL      }      else{          t=head;          while(t->next != NULL) //1.Traversing upto the last element              t=t->next;          t->next=(struct node \*)malloc(sizeof(struct node)); //2.Allocating memory to the next of last node          t->next->data=e; //3.copying element to newly created node's data part          t->next->next=NULL;  //4.making next of newly created node as NULL      }  }  void insert\_after\_el(int e){      struct node \*t,\*s;      if(head==NULL)          printf("\nLinked list is Empty.\n");      else{            t=head;          while(t!=NULL && t->data!=e)              t=t->next;          if(t==NULL)              printf("\n\nElement %d NOT FOUND.\n",e);          else if(t->data==e){              printf("\nElement found.\nEnter the element to insert after %d :",e);              int new\_e;              scanf("%d",&new\_e);              s=(struct node \*)malloc(sizeof(struct node));              s->data=new\_e;              s->next=t->next;              t->next=s;          }      }  }  void disp(){      struct node \*t;      if(head==NULL)          printf("\nLinked list is empty.\n");      else{          t=head;          printf("\nDisplaying Created Linked List : \n");          while(t!=NULL){              printf("\t%d",t->data);              t=t->next;          }      }printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'9\_2\_linkedlist\_insert.exe'  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 1  Enter element to insert at the beginning of linked list : 25  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 1  Enter element to insert at the beginning of linked list : 35  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 2  Enter how many elements to insert(n) : 3  Enter elements to insert at the end of linked list :  50  60  70  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 4  Displaying Created Linked List :  35 25 50 60 70  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 3  Enter the element to search : 50  Element found.  Enter the element to insert after 50 :55  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 4  Displaying Created Linked List :  35 25 50 55 60 70  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DISPLAY.  5.EXIT  Enter any choice : 5  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 9\_3** | **Date: 10/09/2022** |
| **Write a program with functions to delete a node**   * 1. **From the beginning of the linked list**   2. **From the end of the linked list**   3. **The node with specified data element** | |

**Source Code:**

|  |
| --- |
| /\*PROGRAM 9\_3 :Write a program with functions to delete a node  a.From the beginning of the linked list  b.From the end of the linked list  c.The node with specified data element  \*/  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  struct node \*head=NULL;  void insert\_at\_begin();  void insert\_at\_end();  void insert\_after\_el();  void del\_from\_begin();  void del\_from\_end();  void del\_specific();  void disp();  int menu();  int main(){      int ch,n,e,i;      for(ch=menu();ch!=8;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter element to insert at the beginning of linked list : ");                  scanf("%d",&e);                  insert\_at\_begin(e);                  break;              case 2:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to insert at the end of linked list : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert\_at\_end(e);                  }                  break;              case 3:                  printf("\nEnter the element to search : ");                  scanf("%d",&e);                  insert\_after\_el(e);                  break;                case 4:                  del\_from\_begin();                  break;              case 5:                  del\_from\_end();                  break;              case 6:                  printf("\nEnter the  element to delete :  ");                  scanf("%d",&e);                  del\_specific(e);                  break;              case 7:                  disp();                  break;              case 8:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;          }      }      return 0;  }  int menu()  {      int ch;      printf("\nLINKED LIST OPERATIONS->\n1.INSERT AT BEGINNING.\n2.INSERT AT END.\n3.INSERT AFTER A SPECIFIED ELEMENT.");      printf("\n4.DELETE from beginning .\n5.DELETE from end.\n6.DELETE node with specific element.\n7.DISPLAY.\n8.EXIT");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void insert\_at\_begin(int e){      struct node \*t;      t=(struct node \*)malloc(sizeof(struct node));      t->data=e;      t->next=head;      head=t;  }  void insert\_at\_end(int e){      struct node \*t; //creating another pointer for traversing to last element      if(head==NULL){          head=(struct node \*)malloc(sizeof(struct node));  //1.allocating memory to head          head->data=e; //2.copying element to head->data          head->next=NULL; //3.making next of head as NULL      }      else{          t=head;          while(t->next != NULL) //1.Traversing upto the last element              t=t->next;          t->next=(struct node \*)malloc(sizeof(struct node)); //2.Allocating memory to the next of last node          t->next->data=e; //3.copying element to newly created node's data part          t->next->next=NULL;  //4.making next of newly created node as NULL      }  }  void insert\_after\_el(int e){      struct node \*t,\*s;      if(head==NULL)          printf("\nLinked list is Empty.\n");      else{            t=head;          while(t!=NULL && t->data!=e)              t=t->next;          if(t==NULL)              printf("\n\nElement %d NOT FOUND.\n",e);          else if(t->data==e){              printf("\nElement found.\nEnter the element to insert after %d :",e);              int new\_e;              scanf("%d",&new\_e);              s=(struct node \*)malloc(sizeof(struct node));              s->data=new\_e;              s->next=t->next;              t->next=s;          }      }  }  void del\_from\_begin(){      if(head==NULL)          printf("\nLinked List is Empty.\n");        else{          printf("\nSUCCESSFULLY DELETED element %d\n\n",head->data);          head=head->next;      }  }  void del\_from\_end(){      struct node \*t;      if(head==NULL)          printf("\nLinked list is Empty.\n");      else if(head->next==NULL){          printf("\nSUCCESSFULLY DELETED element %d\n\n",head->data);          head=NULL;      }      else{          t=head;          while(t->next->next!=NULL){              t=t->next;          }          printf("\nSUCCESSFULLY DELETED element %d\n\n",t->next->data);          t->next=NULL;        }  }  void del\_specific(int e){      struct node \*t;      if(head==NULL)          printf("\nLinked list is Empty.\n");      else if(head->data==e){          printf("\nSUCCESSFULLY DELETED element %d\n\n",e);          head=head->next;      }      else{          t=head;          while(t->next!=NULL && t->next->data!=e){              t=t->next;          }          if(t->next==NULL)              printf("\nElement NOT FOUND .\n");          else{              printf("\nSUCCESSFULLY DELETED element %d\n\n",e);              t->next=t->next->next;          }      }    }  void disp(){      struct node \*t;      if(head==NULL)          printf("\nLinked list is empty.\n");      else{          t=head;          printf("\nDisplaying Created Linked List : \n");          while(t!=NULL){              printf("\t%d",t->data);              t=t->next;          }      }printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'9\_3\_linkedlist\_deletion.exe'  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 2  Enter how many elements to insert(n) : 5  Enter elements to insert at the end of linked list :  10  20  30  40  50  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 7  Displaying Created Linked List :  10 20 30 40 50  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 4  SUCCESSFULLY DELETED element 10  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 5  SUCCESSFULLY DELETED element 50  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 7  Displaying Created Linked List :  20 30 40  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 6  Enter the element to delete : 30  SUCCESSFULLY DELETED element 30  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 7  Displaying Created Linked List :  20 40  LINKED LIST OPERATIONS->  1.INSERT AT BEGINNING.  2.INSERT AT END.  3.INSERT AFTER A SPECIFIED ELEMENT.  4.DELETE from beginning .  5.DELETE from end.  6.DELETE node with specific element.  7.DISPLAY.  8.EXIT  Enter any choice : 8  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 9\_4** | **Date: 11/09/2022** |
| **Write a program to create a singly linked list of n nodes and display it in reverse order.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 9\_4 :Write a program to create a singly linked list of n nodes and display it in reverse order.  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  struct node \*head=NULL;  void insert();  void reverse\_list();  void disp();  int menu();  int main(){      int ch,n,e,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to insert at the end of linked list : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert(e);                  }                  break;              case 2:                  disp();                  break;              case 3:                  reverse\_list();                  break;              case 4:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nLINKED LIST OPERATIONS->\n1.INSERT.\n2.DISPLAY.\n3.REVERSE LIST.\n4.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void insert(int e){      struct node \*t; //creating another pointer for traversing to last element      if(head==NULL){          head=(struct node \*)malloc(sizeof(struct node));  //1.allocating memory to head          head->data=e; //2.copying element to head->data          head->next=NULL; //3.making next of head as NULL      }      else{          t=head;          while(t->next != NULL) //1.Traversing upto the last element              t=t->next;          t->next=(struct node \*)malloc(sizeof(struct node)); //2.Allocating memory to the next of last node          t->next->data=e; //3.copying element to newly created node's data part          t->next->next=NULL;  //4.making next of newly created node as NULL      }  }  void reverse\_list(){      struct node \*temp=NULL;      struct node \*temp2=NULL;      if(head==NULL)      {          printf("\nLinked List is empty.Reversing NOT POSSIBLE.");      }      else{          while(head!=NULL){              temp2=head->next;              head->next=temp;              temp=head;              head=temp2;          }          head=temp; //At last head will be null and temp will be having address of last node.So make head as temp.          printf("\nLinked List Reversed Successfully.\n");      }  }  void disp(){      struct node \*t;      if(head==NULL)          printf("\nLinked list is empty.\n");      else{          t=head;          printf("\nDisplaying Created Linked List : \n");          while(t!=NULL){              printf("\t%d",t->data);              t=t->next;          }      }printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'9\_4\_linkedlist\_reverse\_display.exe'  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.REVERSE LIST.  4.EXIT.  Enter any choice : 1  Enter how many elements to insert(n) : 5  Enter elements to insert at the end of linked list :  10  20  30  40  50  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.REVERSE LIST.  4.EXIT.  Enter any choice : 2  Displaying Created Linked List :  10 20 30 40 50  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.REVERSE LIST.  4.EXIT.  Enter any choice : 3  Linked List Reversed Successfully.  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.REVERSE LIST.  4.EXIT.  Enter any choice : 2  Displaying Created Linked List :  50 40 30 20 10  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.REVERSE LIST.  4.EXIT.  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 9\_5** | **Date: 11/09/2022** |
| **Sort the elements in a linked list using**  **A.changing the values (swapping the values)**  **B.Changing the address (Swapping the address)** | |

**Source Code:**

|  |
| --- |
| /\*PROGRAM 9\_5 :Sort the elements in a linked list using  a.changing the values (swapping the values)  b.Changing the address (Swapping the address)  \*/  #include <stdio.h>  #include <malloc.h>  struct node  {      int data;      struct node \*next;  };  struct node \*head = NULL;  void insert();  void sort\_by\_swap\_values();  void sort\_by\_swap\_address();  void disp();  int menu();  int main()  {      int ch, n, e, i;      for (ch = menu(); ch != 5; ch = menu())      {          switch (ch)          {          case 1:              printf("\nEnter how many elements to insert(n) : ");              scanf("%d", &n);              printf("\nEnter elements to insert at the end of linked list : \n");              for (i = 0; i < n; i++)              {                  scanf("%d", &e);                  insert(e);              }              break;          case 2:              disp();              break;          case 3:              sort\_by\_swap\_values();              break;          case 4:              sort\_by\_swap\_address();              break;          case 5:              printf("\nExiting.....\n");              break;          default:              printf("\nInvalid Input.TRY AGAIN\n");              break;          }      }      return 0;  }  int menu()  {      int ch;      printf("\nLINKED LIST OPERATIONS->\n1.INSERT.\n2.DISPLAY.\n3.SORT BY SWAPPING VALUES.\n4.SORT BY SWAPPING ADDRESS\n5.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d", &ch);      return ch;  }  void insert(int e)  {      struct node \*t; // creating another pointer for traversing to last element      if (head == NULL)      {          head = (struct node \*)malloc(sizeof(struct node)); // 1.allocating memory to head          head->data = e;                                    // 2.copying element to head->data          head->next = NULL;                                 // 3.making next of head as NULL      }      else      {          t = head;          while (t->next != NULL) // 1.Traversing upto the last element              t = t->next;          t->next = (struct node \*)malloc(sizeof(struct node)); // 2.Allocating memory to the next of last node          t->next->data = e;                                    // 3.copying element to newly created node's data part          t->next->next = NULL;                                 // 4.making next of newly created node as NULL      }  }  void sort\_by\_swap\_values()  {      struct node \*i, \*j; // Creating two pointers to structure.      int temp;      if (head == NULL)      {          printf("\nLinked list is Empty.Sorting NOT POSSIBLE.");      }      else      {          for (i = head; i != NULL; i = i->next)          {              for (j = i; j != NULL; j = j->next)              {                  if (i->data > j->data)                  {                      temp = i->data;                      i->data = j->data;                      j->data = temp;                  }              }          }          printf("\nSorting Successfull completed.\n");      }  }  void sort\_by\_swap\_address()  {      struct node \*i = head;      struct node \*t1 = head;      while (i->next != NULL)      {          struct node \*t2 = i;          struct node \*j = i->next;          while (j != NULL)          {              if (i->data > j->data)              {                  t2->next = j->next;                  if (head == i)                      t1 = head = j;                  else                      t1->next = j;                  j->next = i;                  i = t1;              }              else                  t2 = j;              j = j->next;          }          t1 = i;          i = i->next;      }      printf("\nSorting successfully completed by swapping addresses.\n");  }  void disp()  {      struct node \*t;      if (head == NULL)          printf("\nLinked list is empty.\n");      else      {          t = head;          printf("\nDisplaying Created Linked List : \n");          while (t != NULL)          {              printf("\t%d", t->data);              t = t->next;          }      }      printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'9\_5\_linkedlist\_sort\_element.exe'  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 1  Enter how many elements to insert(n) : 5  Enter elements to insert at the end of linked list :  25  13  2  22  31  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 2  Displaying Created Linked List :  25 13 2 22 31  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 4  Sorting successfully completed by swapping addresses.  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 2  Displaying Created Linked List :  2 13 22 25 31  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 1  Enter how many elements to insert(n) : 3  Enter elements to insert at the end of linked list :  5  8  29  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 2  Displaying Created Linked List :  2 13 22 25 31 5 8 29  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 3  Sorting Successfull completed.  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 2  Displaying Created Linked List :  2 5 8 13 22 25 29 31  LINKED LIST OPERATIONS->  1.INSERT.  2.DISPLAY.  3.SORT BY SWAPPING VALUES.  4.SORT BY SWAPPING ADDRESS  5.EXIT.  Enter any choice : 5  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 10\_1** | **Date: 15/09/2022** |
| **Polynomial using linked list - addition and multiplication.** | |

**Source Code:**

|  |
| --- |
| // PROGRAM 10\_1 :Polynomial using linked list - addition and multiplication.  #include <stdio.h>  #include <stdlib.h>  struct node  {      float coeff;      int exp;      struct node \*next;  };  typedef struct node node;  struct node \*sort(struct node \*head, float co, int ex)  {      struct node \*temp;      struct node \*newP = malloc(sizeof(struct node));      newP->coeff = co;      newP->exp = ex;      newP->next = NULL;      if (head == NULL || ex > head->exp)      {          newP->next = head;          head = newP;      }      else      {          temp = head;          while (temp->next != NULL && temp->next->exp >= ex)              temp = temp->next;          newP->next = temp->next;          temp->next = newP;      }      return head;  }  struct node \*poly(struct node \*head)  {      int n, i;      float coeff;      int expo;      printf("\nEnter the number of terms: ");      scanf("%d", &n);      for (i = 0; i < n; i++)      {          printf("Enter the coefficient for term %d: ", i + 1);          scanf("%f", &coeff);          printf("Enter the exponent for term %d: ", i + 1);          scanf("%d", &expo);          head = sort(head, coeff, expo);      }      return head;  }  void disp(struct node \*head)  {      if (head == NULL)          printf("No Polynomial.");      else      {          struct node \*temp;          for (temp = head; temp->next != NULL; temp = temp->next)          {              printf("(%.0fx^%d)+", temp->coeff, temp->exp);          }          if (temp->exp == 0)              printf("%.0f", temp->coeff);          printf("\n");      }  }  void polyAdd(struct node \*head1, struct node \*head2)  {      struct node \*ptr1 = head1;      struct node \*ptr2 = head2;      struct node \*sum = NULL;      while (ptr1 != NULL && ptr2 != NULL)      {          if (ptr1->exp == ptr2->exp)          {              sum = sort(sum, ptr1->coeff + ptr2->coeff, ptr1->exp);              ptr1 = ptr1->next;              ptr2 = ptr2->next;          }          else if (ptr1->exp > ptr2->exp)          {              sum = sort(sum, ptr1->coeff, ptr1->exp);              ptr1 = ptr1->next;          }          else if (ptr1->exp < ptr2->exp)          {              sum = sort(sum, ptr2->coeff, ptr2->exp);              ptr2 = ptr2->next;          }      }      while (ptr1 != NULL)      {          sum = sort(sum, ptr1->coeff, ptr1->exp);          ptr1 = ptr1->next;      }      while (ptr2 != NULL)      {          sum = sort(sum, ptr2->coeff, ptr2->exp);          ptr2 = ptr2->next;      }      printf("\nAdded polynomial is: ");      disp(sum);  }  node \*polyMult(node \*head1, node \*head2, node \*pro)  {      node \*ptr1 = head1;      node \*ptr2 = head2;      if (head1 == NULL || head2 == NULL)      {          printf("\nNo polynomial\n");          return NULL;      }      while (ptr1 != NULL)      {          while (ptr2 != NULL)          {              float coeffPro = ptr1->coeff \* ptr2->coeff;              int expoSum = ptr1->exp + ptr2->exp;              pro = sort(pro, coeffPro, expoSum);              ptr2 = ptr2->next;          }          ptr1 = ptr1->next;          ptr2 = head2;      }      return pro;  }  node \*add(node \*pro, node \*res)  {      node \*temp1, \*temp2;      temp1 = pro;      while (temp1->next != NULL)      {          temp2 = temp1->next;          while (temp2 != NULL)          {              if (temp1->exp == temp2->exp)              {                  float coeffSum = temp1->coeff + temp2->coeff;                  res = sort(res, coeffSum, temp1->exp);                  temp1 = temp1->next;                  break;              }              else              {                  res = sort(res, temp1->coeff, temp1->exp);                  break;              }              temp2 = temp2->next;          }          temp1 = temp1->next;      }      res = sort(res, temp1->coeff, temp1->exp);      return res;  }  int main()  {      struct node \*head1 = NULL;      struct node \*head2 = NULL;      node \*pro = NULL;      node \*res = NULL;      printf("\nEnter the First polynomial details--->\n ");      head1 = poly(head1);      printf("\nEnter the second polynomial--->\n ");      head2 = poly(head2);      disp(head1);      disp(head2);      polyAdd(head1, head2);      pro = polyMult(head1, head2, pro);      node \*res2 = NULL;      res = add(pro, res);      res2 = add(res, res2);      printf("\nProduct is: ");      disp(res2);      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'10\_1\_poly\_linkedlist\_add\_mul.exe'  Enter the First polynomial details--->  Enter the number of terms: 3  Enter the coefficient for term 1: 2  Enter the exponent for term 1: 2  Enter the coefficient for term 2: 1  Enter the exponent for term 2: 1  Enter the coefficient for term 3: 1  Enter the exponent for term 3: 0  Enter the second polynomial--->  Enter the number of terms: 3  Enter the coefficient for term 1: 1  Enter the exponent for term 1: 2  Enter the coefficient for term 2: 3  Enter the exponent for term 2: 1  Enter the coefficient for term 3: 2  Enter the exponent for term 3: 0  (2x^2)+(1x^1)+1  (1x^2)+(3x^1)+2  Added polynomial is: (3x^2)+(4x^1)+3  Product is: (2x^4)+(7x^3)+(8x^2)+(5x^1)+2  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 10\_2** | **Date: 17/09/2022** |
| **Linked list using names - insert, delete, display, sort, reverse, count.** | |

**Source Code:**

|  |
| --- |
| // PROGRAM 10\_2 :Linked list using names - insert, delete, display, sort, reverse, count.  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  struct node  {      char data[20];      struct node \*next;  };  typedef struct node list;  list \*head = NULL;  void insert\_l(char e[20])  {      list \*t;      if (head == NULL)      {          head = (list \*)malloc(sizeof(list));          strcpy(head->data, e);          head->next = NULL;      }      else      {          t = head;          while (t->next != NULL)          {              t = t->next;          }          t->next = (list \*)malloc(sizeof(list));          strcpy(t->next->data, e);          t->next->next = NULL;      }  }  void display()  {      printf("\nDisplaying Names : ");      list \*t;      if (head == NULL)      {          printf("List is empty");      }      else      {          t = head;          while (t != NULL)          {              printf("\t%s", t->data);              t = t->next;          }          printf("\n");      }  }  void count()  {      list \*t;      int c = 0;      if (head == NULL)      {          printf("List is empty");      }      else      {          t = head;          while (t != NULL)          {              c++;              t = t->next;          }          printf("\nCount %d\n", c);      }  }  void delete\_l(char e[20])  {      list \*t;      if (head == NULL)      {          printf("List is empty");      }      else if (strcmp(head->data, e) == 0)      {          printf("\nDeleted : %s\n", head->data);          head = head->next;      }      else      {          t = head;          while (t->next != NULL && strcmp(t->next->data, e) != 0)          {              t = t->next;          }          if (t->next == NULL)          {              printf("\nNot found");          }          else if (strcmp(t->next->data, e) == 0)          {              printf("\nDeleted %s\n", t->next->data);              t->next = t->next->next;          }      }  }  void reverseList()  {      list \*prevNode, \*curNode;      if (head != NULL)      {          prevNode = head;          curNode = head->next;          head = head->next;          prevNode->next = NULL;          while (head != NULL)          {              head = head->next;              curNode->next = prevNode;              prevNode = curNode;              curNode = head;          }          head = prevNode;      }      printf("\nReversed names are : ");      display();  }  void sortList()  {      list \*current = head, \*index = NULL;      char temp[20];      if (head == NULL)      {          return;      }      else      {          while (current != NULL)          {              index = current->next;              while (index != NULL)              {                  if (strcmp(current->data, index->data) > 0)                  {                      strcpy(temp, current->data);                      strcpy(current->data, index->data);                      strcpy(index->data, temp);                  }                  index = index->next;              }              current = current->next;          }      }      printf("\nNames sorted successfully.\n");  }  int main()  {      int ch = 0;      char e[20], key[20];      for (; ch != 7;)      {          printf("\nLinked List using Names Operations:\n1.Insert Name\n2.Delete\n3.Display \n4.Count\n5.Reverse\n6.Sort list\n7.Exit");          printf("\nEnter your choice: ");          scanf("%d", &ch);          switch (ch)          {          case 1:              fflush(stdin);              printf("\nEnter the element to be inserted: ");              gets(e);              insert\_l(e);              break;          case 2:              fflush(stdin);              printf("\nEnter the element to be delete: ");              gets(key);              delete\_l(key);              break;          case 3:              display();              break;          case 4:              count();              break;          case 5:              reverseList();              break;          case 6:              sortList();              break;          case 7:              break;          default:              printf("\nInvalid input.try again.");          }      }      return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'10\_2\_poly\_linkedlist\_sort\_rev\_count.exe'  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 1  Enter the element to be inserted: Nihal  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 1  Enter the element to be inserted: Jibin  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 1  Enter the element to be inserted: Abhinav  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 1  Enter the element to be inserted: Majo  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 3  Displaying Names : Nihal Jibin Abhinav Majo  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 4  Count 4  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 5  Reversed names are :  Displaying Names : Majo Abhinav Jibin Nihal  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 6  Names sorted successfully.  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 3  Displaying Names : Abhinav Jibin Majo Nihal  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 2  Enter the element to be delete: Jibin  Deleted Jibin  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 3  Displaying Names : Abhinav Majo Nihal  Linked List using Names Operations:  1.Insert Name  2.Delete  3.Display  4.Count  5.Reverse  6.Sort list  7.Exit  Enter your choice: 7  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 11\_1\_a** | **Date: 20/09/2022** |
| **Linked Stack** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 11\_1\_a : Linked Stack  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  typedef struct node stack;  stack \*top=NULL; //pointer top pointing to structure Stack.  void push\_to\_linkedstack();  void pop\_from\_linkedstack();  void peek\_from\_linkedstack();  void disp();  int menu();  int main(){  int ch,n,e,i;      for(ch=menu();ch!=5;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to PUSH(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to PUSH to linked Stack : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      push\_to\_linkedstack(e);                  }                  break;              case 2:                  pop\_from\_linkedstack();                  break;              case 3:                  peek\_from\_linkedstack();                  break;              case 4:                  disp();                  break;              case 5:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nLINKED STACK OPERATIONS->\n1.PUSH to Linked Stack.\n2.POP from Linked Stack.\n3.PEEK from Linked stack\n4.DISPLAY.\n5.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void push\_to\_linkedstack(int e){      stack \*t; //creating another pointer to structure stack      t=(stack \*)malloc(sizeof(stack)); //allocating memory to new node      t->data=e; //copying data to new node      t->next=top; //make the next of new node as current top.      top=t; //assign the address of new node to top.  }  void pop\_from\_linkedstack(){      if(top==NULL){          printf("\nLinked Stack is Empty.Cannot perform POP\n");      }      else{          printf("\nPoped element %d.\n",top->data);          top=top->next;      }  }  void peek\_from\_linkedstack(){      if(top==NULL){          printf("\nLinked Stack is Empty.Cannot perform PEEK\n");      }      else{          printf("\nPEEKed element is  %d.\n",top->data);      }  }  void disp(){      struct node \*t;      if(top==NULL)          printf("\nLinked Stack is empty.\n");      else{          t=top;          printf("\nDisplaying Created Linked Stack : \n");          while(t!=NULL){              printf("\t%d",t->data);              t=t->next;          }      }printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'11\_1\_a\_linked\_stack.exe'  LINKED STACK OPERATIONS->  1.PUSH to Linked Stack.  2.POP from Linked Stack.  3.PEEK from Linked stack  4.DISPLAY.  5.EXIT.  Enter any choice : 2  Linked Stack is Empty.Cannot perform POP  LINKED STACK OPERATIONS->  1.PUSH to Linked Stack.  2.POP from Linked Stack.  3.PEEK from Linked stack  4.DISPLAY.  5.EXIT.  Enter any choice : 1  Enter how many elements to PUSH(n) : 5  Enter elements to PUSH to linked Stack :  10  20  30  40  50  LINKED STACK OPERATIONS->  1.PUSH to Linked Stack.  2.POP from Linked Stack.  3.PEEK from Linked stack  4.DISPLAY.  5.EXIT.  Enter any choice : 3  PEEKed element is 50.  LINKED STACK OPERATIONS->  1.PUSH to Linked Stack.  2.POP from Linked Stack.  3.PEEK from Linked stack  4.DISPLAY.  5.EXIT.  Enter any choice : 2  Poped element 50.  LINKED STACK OPERATIONS->  1.PUSH to Linked Stack.  2.POP from Linked Stack.  3.PEEK from Linked stack  4.DISPLAY.  5.EXIT.  Enter any choice : 3  PEEKed element is 40.  LINKED STACK OPERATIONS->  1.PUSH to Linked Stack.  2.POP from Linked Stack.  3.PEEK from Linked stack  4.DISPLAY.  5.EXIT.  Enter any choice : 5  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 11\_1\_b** | **Date: 21/09/2022** |
| **Linked Queue.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 11\_1\_b : Linked Queue  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  typedef struct node queue;  enum op{enqueue=1,dequeue,display,EXIT};//enum is used to assign names to integral constants  queue \*f=NULL,\*r=NULL;  void enqueue\_to\_linkedqueue();  void dequeue\_from\_linkedqueue();  void disp();  int menu();  int main(){      int ch,e;      for(ch=menu();ch!=EXIT;ch=menu()){          switch(ch){              case enqueue:                      printf("\nEnter the element to enqueue: ");                      scanf("%d",&e);                      enqueue\_to\_linkedqueue(e);                      break;              case dequeue:                      dequeue\_from\_linkedqueue();                      break;              case display:                      disp();                      break;              case EXIT:                      printf("\nExiting......");                      break;              default:                      printf("\nInvalid Input.Try Again.\n");          }      }      return 0;  }  int menu(){      int ch;      printf("\nLinked Queue Operations ->\n1.ENQUEUE.\n2.DEQUEUE.\n3.DISPLAY.\n4.EXIT.\n");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void enqueue\_to\_linkedqueue(int e){      queue \*t=(queue \*)malloc(sizeof(queue));//allocate a new node      t->data=e;//copy the data to new node      t->next=NULL;//make next of new node as NULL      if(f==NULL){          //assign new node address to f and r          f=t;          r=t;      }      else{          r->next=t;          r=t;      }  }  void dequeue\_from\_linkedqueue(){      if(f==NULL){          printf("\nQueue is empty.Dequeue not possible .\n");      }      else{          printf("\nDequeued %d",f->data);          f=f->next;          if(f==NULL){              r=NULL;          }      }  }  void disp(){      struct node \*t;      if(f==NULL)          printf("\nLinked Queue is empty.\n");      else{          t=f;          printf("\nDisplaying Created Linked Queue : \n");          while(t!=NULL){              printf("\t%d",t->data);              t=t->next;          }      }printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'11\_1\_b\_linked\_queue.exe'  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Queue is empty.Dequeue not possible .  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter the element to enqueue: 20  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter the element to enqueue: 30  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter the element to enqueue: 35  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Linked Queue :  20 30 35  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Dequeued 20  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Linked Queue :  30 35  Linked Queue Operations ->  1.ENQUEUE.  2.DEQUEUE.  3.DISPLAY.  4.EXIT.  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 11\_1\_c** | **Date: 22/09/2022** |
| **Circular Linked List.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 11\_1\_c : Circular Linked List  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;  };  typedef struct node list;  list \*head=NULL;  void insert\_to\_cll();  void delete\_from\_cll();  void disp();  int menu();  int main(){  int ch,n,e,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to INSERT(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to INSERT to Circular linked List : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert\_to\_cll(e);                  }                  break;              case 2:                  printf("\nEnter the element to DELETE : ");                  scanf("%d",&e);                  delete\_from\_cll(e);                  break;              case 3:                  disp();                  break;              case 4:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nCIRCULAR LINKED LIST OPERATIONS->\n1.INSERT to CLL.\n2.Delete from CLL.\n3.DISPLAY.\n4.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void insert\_to\_cll(int e){      list \*t; //pointer to structure.      if(head==NULL){          head=(list \*)malloc(sizeof(list)); //allocating memory to HEAD          head->data=e;//copy data          head->next=head;//make next of head as head ,to make a circular link      }      else{          t=head;          while(t->next!=head){//traversing until t->next becomes head ,since it is CLL              t=t->next;              }          t->next=(list \*)malloc(sizeof(list)); //allocating memory to new node          t->next->data=e; //copying data to new node          t->next->next=head; //making next of new node as head      }    }  void delete\_from\_cll(int e){      list \*t;      if(head==NULL){ //case\_1:list is empty          printf("\nCircular Linked List is Empty.Cannot perform Deletion\n");      }      else if(head->data==e && head->next==head){//case\_2:Delete first element,when there is only one element          printf("\nDeleted element %d.\n",head->data);          head=NULL;      }      else if(head->data==e){//case\_3:Delete first element,when there is more than one element          t=head;          while(t->next!=head){//traversing              t=t->next;          }          printf("\nDeleted element %d.\n",head->data);          t->next=head->next; //updating last node address to next of head          head=head->next;      }      else{          t=head;          while(t->next!=head &&t->next->data!=e){              t=t->next;          }          if(t->next==head) //case\_4:Element Not found              printf("\nElement %d NOT FOUND.Deletion Failed\n",e);          else{              printf("\nDeleted element %d.\n",t->next->data);              t->next=t->next->next;//case\_5:Delete intermediate or Last element          }      }  }  //DISPLAY\_2  void disp(){      list \*t;      if(head==NULL)          printf("\nCircular Linked List is empty.\n");      else{          t=head;          printf("\nDisplaying Created Circular Linked List : \n");          do{              printf("\t%d",t->data);              t=t->next;          }while(t!=head);      }  }  // void disp(){  //  list \*t;  //  if(head==NULL)  //      printf("\nCircular Linked List is empty.\n");  //  else{  //      t=head;  //         printf("\nDisplaying Created Circular Linked List : \n");  //      while(t->next!=head){ //traversing until t->next becomes head  //          printf("\t%d",t->data);  //          t=t->next;  //      }  //  }printf("\t%d\n",t->data); //To display the last element.  // } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'11\_1\_c\_circular\_linked\_list.exe'  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Enter the element to DELETE : 10  Circular Linked List is Empty.Cannot perform Deletion  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter how many elements to INSERT(n) : 5  Enter elements to INSERT to Circular linked List :  10  20  30  40  50  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Circular Linked List :  10 20 30 40 50  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Enter the element to DELETE : 30  Deleted element 30.  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Enter the element to DELETE : 50  Deleted element 50.  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Circular Linked List :  10 20 40  CIRCULAR LINKED LIST OPERATIONS->  1.INSERT to CLL.  2.Delete from CLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 11\_1\_d** | **Date: 23/09/2022** |
| **Circular Linked Queue.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 11\_1\_d : Circular Linked Queue  #include <stdio.h>  #include <stdlib.h>  struct node {      int data;      struct node \*next;  };  typedef struct node queue;  queue \*f=NULL,\*r=NULL;  void enqueue\_to\_clq();  void dequeue\_from\_clq();  void disp();  int menu();  int main(){  int ch,n,e,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to ENQUEUE(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to enqueue to Circular linked Queue : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      enqueue\_to\_clq(e);                  }                  break;              case 2:                  dequeue\_from\_clq();                  break;              case 3:                  disp();                  break;              case 4:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nCIRCULAR LINKED QUEUE OPERATIONS->\n1.ENQUEUE to CLQ.\n2.DEQUEUE from CLQ.\n3.DISPLAY.\n4.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void enqueue\_to\_clq(int e){      queue \*t;      if(f==NULL){          f=(queue \*)malloc(sizeof(queue)); //allocating memory to HEAD          f->data=e;//copy data          f->next=f;          r=f;      }      else{          t=(queue \*)malloc(sizeof(queue));          t->data=e;          t->next=f;          r->next=t;          r=t;      }    }  void dequeue\_from\_clq(){      if(f==NULL){ //case\_1:empty queue          printf("\nQueue is empty.Dequeue not possible .\n");      }      else if(f->next==f){//case\_2:dequeue first element when there is only one element          printf("\nDequeued %d\n",f->data);          f=r=NULL;      }      else{          //case\_3:dequeue first element when there are several elements          printf("\nDequeued %d\n",f->data);          f=f->next;          r->next=f;      }  }  void disp(){      struct node \*t;      if(f==NULL)          printf("\nCircular Linked Queue is empty.\n");      else{          t=f;          printf("\nDisplaying Created Circular Linked Queue : \n");          do{              printf("\t%d",t->data);              t=t->next;          }while(t!=f);      }printf("\n");  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'11\_1\_d\_circular\_linked\_queue.exe'  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Queue is empty.Dequeue not possible .  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter how many elements to ENQUEUE(n) : 5  Enter elements to enqueue to Circular linked Queue :  10  20  30  40  50  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Circular Linked Queue :  10 20 30 40 50  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Dequeued 10  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Dequeued 20  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Circular Linked Queue :  30 40 50  CIRCULAR LINKED QUEUE OPERATIONS->  1.ENQUEUE to CLQ.  2.DEQUEUE from CLQ.  3.DISPLAY.  4.EXIT.  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 11\_1\_e** | **Date: 25/09/2022** |
| **Doubly Linked List.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 11\_1\_e :Doubly linked list  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;      struct node \*prev;  };  typedef struct node list;  list \*head=NULL;  void insert\_to\_dll();  void delete\_from\_dll();  void disp();  int menu();  int main(){  int ch,n,e,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to INSERT(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to INSERT to Doubly linked List : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert\_to\_dll(e);                  }                  break;              case 2:                  printf("\nEnter the element to DELETE : ");                  scanf("%d",&e);                  delete\_from\_dll(e);                  break;              case 3:                  disp();                  break;              case 4:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nDOUBLY LINKED LIST OPERATIONS->\n1.INSERT to DLL.\n2.Delete from DLL.\n3.DISPLAY.\n4.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void insert\_to\_dll(int e){      list \*t; //pointer to structure.      if(head==NULL){          head=(list \*)malloc(sizeof(list)); //allocating new node to HEAD          head->data=e;//copy data          head->next=NULL;//make next of new node as NULL          head->prev=NULL;//make prev of new node as NULL      }      else{          t=head;          while(t->next!=NULL){//traversing              t=t->next;              }          t->next=(list \*)malloc(sizeof(list)); //allocating memory to new node          t->next->data=e; //copying data          t->next->next=NULL; //make next of new node as NULL          t->next->prev=t;//make the prev of new node as current last node      }    }  void delete\_from\_dll(int e){      list \*t;      if(head==NULL){ //case\_1:list is empty          printf("\nDoubly Linked List is Empty.Cannot perform Deletion\n");      }      else if(head->data==e && head->next==NULL){//case\_2:Delete first element,when there is only one element          printf("\nDeleted element %d.\n",head->data);          head=NULL;      }      else if(head->data==e){//case\_3:Delete first element,when there is more than one element          printf("\nDeleted element %d.\n",head->data);          //here we no need to traverse          head=head->next; //moving head to second element          head->prev=NULL; //making prev as NULL      }      else{          t=head->next;          while(t!=NULL &&t->data!=e){              t=t->next;          }          if(t==NULL) //case\_4:Element Not found              printf("\nElement %d NOT FOUND.Deletion Failed\n",e);          else if(t->next==NULL){              //case\_5:Delete Last element              printf("\nDeleted element %d.\n",t->data);              t->prev->next=t->next;          }else{              //case\_6:Delete Intermediate element              printf("\nDeleted element %d.\n",t->data);              t->prev->next=t->next;              t->next->prev=t->prev;          }      }  }  void disp(){      list \*t;      if(head==NULL)          printf("\nDoubly Linked List is empty.\n");      else{          t=head;          printf("\nDisplaying Created Doubly Linked List : \n");          while(t!=NULL){ //traversing              printf("\t%d",t->data);              t=t->next;          }      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'11\_1\_e\_doubly\_linked\_list.exe'  DOUBLY LINKED LIST OPERATIONS->  1.INSERT to DLL.  2.Delete from DLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter how many elements to INSERT(n) : 5  Enter elements to INSERT to Doubly linked List :  10  20  30  40  50  DOUBLY LINKED LIST OPERATIONS->  1.INSERT to DLL.  2.Delete from DLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Doubly Linked List :  10 20 30 40 50  DOUBLY LINKED LIST OPERATIONS->  1.INSERT to DLL.  2.Delete from DLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Enter the element to DELETE : 30  Deleted element 30.  DOUBLY LINKED LIST OPERATIONS->  1.INSERT to DLL.  2.Delete from DLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Doubly Linked List :  10 20 40 50  DOUBLY LINKED LIST OPERATIONS->  1.INSERT to DLL.  2.Delete from DLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 11\_1\_f** | **Date: 26/09/2022** |
| **Circular Doubly Linked List.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 11\_1\_f :Circular Doubly linked list  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*next;      struct node \*prev;  };  typedef struct node list;  list \*head=NULL;  void insert\_to\_cdll();  void delete\_from\_cdll();  void disp();  int menu();  int main(){  int ch,n,e,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to INSERT(n) : ");                  scanf("%d",&n);                  printf("\nEnter elements to INSERT to Circular Doubly linked List : \n");                  for(i=0;i<n;i++){                      scanf("%d",&e);                      insert\_to\_cdll(e);                  }                  break;              case 2:                  printf("\nEnter the element to DELETE : ");                  scanf("%d",&e);                  delete\_from\_cdll(e);                  break;              case 3:                  disp();                  break;              case 4:                  printf("\nExiting.....\n");                  break;              default:                  printf("\nInvalid Input.TRY AGAIN\n");                  break;            }      }      return 0;  }  int menu()  {      int ch;      printf("\nCIRCULAR DOUBLY LINKED LIST OPERATIONS->\n1.INSERT to CDLL.\n2.Delete from CDLL.\n3.DISPLAY.\n4.EXIT.");      printf("\n\nEnter any choice : ");      scanf("%d",&ch);      return ch;  }  void insert\_to\_cdll(int e){      list \*t; //pointer to structure.      if(head==NULL){          head=(list \*)malloc(sizeof(list)); //allocating new node to HEAD          head->data=e;//copy data          head->next=head;//make next of new node as head          head->prev=head;//make prev of new node as head      }      else{          t=head;          while(t->next!=head){//traversing until t->next is head              t=t->next;              }          t->next=(list \*)malloc(sizeof(list)); //allocating memory to new node          t->next->data=e; //copying data          t->next->next=head; //make the next of new node as head          t->next->prev=t; //make the prev of new node as current last node(t)          head->prev=t->next;//make the prev of first node as the new node's address      }    }  void delete\_from\_cdll(int e){      list \*t;      if(head==NULL){ //case\_1:list is empty          printf("\nCircular Doubly Linked List is Empty.Cannot perform Deletion\n");      }      else if(head->data==e && head->next==head){//case\_2:Delete first element,when there is only one element          printf("\nDeleted element %d.\n",head->data);          head=NULL;      }      else if(head->data==e){//case\_3:Delete first element,when there is more than one element          printf("\nDeleted element %d.\n",head->data);          //No need traversal          head->prev->next=head->next;          head->next->prev=head->prev;          head=head->next;      }      else{          t=head->next;          while(t!=head &&t->data!=e){//traversing              t=t->next;          }          if(t==head) //case\_4:Element Not found              printf("\nElement %d NOT FOUND.Deletion Failed\n",e);         else{              //case\_5:Delete Intermediate or Last element              printf("\nDeleted element %d.\n",t->data);              t->prev->next=t->next;              t->next->prev=t->prev;          }      }  }  void disp(){      list \*t;      if(head==NULL)          printf("\nCircular Doubly Linked List is empty.\n");      else{          t=head;          printf("\nDisplaying Created Circular Doubly Linked List : \n");          do{              printf("\t%d",t->data);              t=t->next;          }while(t!=head);      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'11\_1\_f\_circular\_doubly\_linked\_list.exe'  CIRCULAR DOUBLY LINKED LIST OPERATIONS->  1.INSERT to CDLL.  2.Delete from CDLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 1  Enter how many elements to INSERT(n) : 5  Enter elements to INSERT to Circular Doubly linked List :  10  20  30  40  50  CIRCULAR DOUBLY LINKED LIST OPERATIONS->  1.INSERT to CDLL.  2.Delete from CDLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Circular Doubly Linked List :  10 20 30 40 50  CIRCULAR DOUBLY LINKED LIST OPERATIONS->  1.INSERT to CDLL.  2.Delete from CDLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 2  Enter the element to DELETE : 20  Deleted element 20.  CIRCULAR DOUBLY LINKED LIST OPERATIONS->  1.INSERT to CDLL.  2.Delete from CDLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 3  Displaying Created Circular Doubly Linked List :  10 30 40 50  CIRCULAR DOUBLY LINKED LIST OPERATIONS->  1.INSERT to CDLL.  2.Delete from CDLL.  3.DISPLAY.  4.EXIT.  Enter any choice : 4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 12\_1** | **Date: 27/09/2022** |
| **Binary search tree insertion and display Traversal using inorder, preorder and postorder using recursion.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 12\_1 : Binary search tree insertion and display Traversal using inorder, preorder and postorder using recursion  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*left;      struct node \*right;  };  typedef struct node tree;  tree \*root;  void insert();  void inorder();  void preorder();  void postorder();  int menu();  int main(){      int ch,e,n,i;      for(ch=menu();ch!=5;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  for(i=0;i<n;i++){                       printf("\nEnter the element to insert : ");                       scanf("%d",&e);                       insert(e);                  }                  break;              case 2:                  printf("\n\nInorder of entered BST is : ");                  inorder(root);                  break;              case 3:                  printf("\n\nPreorder of entered BST is : ");                  preorder(root);                  break;              case 4:                  printf("\n\nPostorder of entered BST is : ");                  postorder(root);                  break;              case 5:                  printf("\nExiting....");                  break;              default:                  printf("\nInvalid Input . Try again.\n");          }      }        return 0;  }  int menu(){      int ch;      printf("\nBST OPERATIONS ->.\n1.INSERT.\n2.INORDER.\n3.PREORDER.\n4.POSTORDER.\n5.EXIT");      printf("\nEnter any choice :");      scanf("%d",&ch);      return ch;  }  void insert(int e){      tree \*p,\*x;      if(root==NULL){          root=(tree \*)malloc(sizeof(tree));          root->data=e;          root->left=NULL;          root->right=NULL;      }      else{          p=root;          while(p!=NULL){              x=p;              if(e < p->data){                  p=p->left;              }              else{                  p=p->right;              }          }          if(e > x->data){              x->right=(tree \*)malloc(sizeof(tree));              x->right->data=e;              x->right->left=NULL;              x->right->right=NULL;          }          else{              x->left=(tree \*)malloc(sizeof(tree));              x->left->data=e;              x->left->left=NULL;              x->left->right=NULL;          }      }  }  void inorder(tree \*r){        if(r!=NULL){          inorder(r->left);          printf("\t%d",r->data);          inorder(r->right);      }  }  void preorder(tree \*r){        if(r!=NULL){          printf("\t%d",r->data);          preorder(r->left);          preorder(r->right);      }  }  void postorder(tree \*r){        if(r!=NULL){          postorder(r->left);          postorder(r->right);          printf("\t%d",r->data);      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'12\_1\_BST\_insert\_display.exe'  BST OPERATIONS ->.  1.INSERT.  2.INORDER.  3.PREORDER.  4.POSTORDER.  5.EXIT  Enter any choice :1  Enter how many elements to insert(n) : 12  Enter the element to insert : 200  Enter the element to insert : 250  Enter the element to insert : 220  Enter the element to insert : 10  Enter the element to insert : 5  Enter the element to insert : 100  Enter the element to insert : 240  Enter the element to insert : 235  Enter the element to insert : 245  Enter the element to insert : 150  Enter the element to insert : 120  BST OPERATIONS ->.  1.INSERT.  2.INORDER.  3.PREORDER.  4.POSTORDER.  5.EXIT  Enter any choice :2  Inorder of entered BST is : 5 10 100 120 150 170 200 220 235 240 245 250  BST OPERATIONS ->.  1.INSERT.  2.INORDER.  3.PREORDER.  4.POSTORDER.  5.EXIT  Enter any choice :3  Preorder of entered BST is : 200 10 5 100 150 120 170 250 220 240 235 245  BST OPERATIONS ->.  1.INSERT.  2.INORDER.  3.PREORDER.  4.POSTORDER.  5.EXIT  Enter any choice :4  Postorder of entered BST is : 5 120 170 150 100 10 235 245 240 220 250 200  BST OPERATIONS ->.  1.INSERT.  2.INORDER.  3.PREORDER.  4.POSTORDER.  5.EXIT  Enter any choice :5 |

|  |  |
| --- | --- |
| **Program # 12\_2** | **Date: 29/09/2022** |
| **Binary search tree insertion and display in-order without using recursion.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 12\_2 : Binary search tree insertion and display in-order without using recursion  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*left;      struct node \*right;  };  typedef struct node tree;  struct stack{      tree \*r;      struct stack \*next;  };  typedef struct stack stack;  tree \*root=NULL;  stack \*top=NULL;  void insert();  void push();  tree \* pop();  void inorder();  void inorder\_wur();  int menu();  int main(){      int ch,e,n,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  for(i=0;i<n;i++){                       printf("\nEnter the element to insert : ");                       scanf("%d",&e);                       insert(e);                  }                  break;              case 2:                  printf("\n\nInorder  of entered BST is : ");                  inorder(root);                  break;              case 3:                  printf("\n\ninorder Without Using Reccursion of entered BST is : ");                  inorder\_wur(root);                  break;              case 4:                  printf("\nExiting....");                  break;              default:                  printf("\nInvalid Input . Try again.\n");          }      }        return 0;  }  int menu(){      int ch;      printf("\nBST OPERATIONS ->.\n1.INSERT.\n2.INORDER(Using reccusrion).\n3.INORDER(Without using reccusrion).\n4.EXIT");      printf("\nEnter any choice :");      scanf("%d",&ch);      return ch;  }  void insert(int e){      tree \*p,\*x;      if(root==NULL){          root=(tree \*)malloc(sizeof(tree));          root->data=e;          root->left=NULL;          root->right=NULL;      }      else{          p=root;          while(p!=NULL){              x=p;              if(e < p->data){                  p=p->left;              }              else{                  p=p->right;              }          }          if(e > x->data){              x->right=(tree \*)malloc(sizeof(tree));              x->right->data=e;              x->right->left=NULL;              x->right->right=NULL;          }          else{              x->left=(tree \*)malloc(sizeof(tree));              x->left->data=e;              x->left->left=NULL;              x->left->right=NULL;          }      }  }  void push(tree \*t){      stack \*temp=(stack \*)malloc(sizeof(stack));      temp->r=t;      temp->next=top;      top=temp;  }  tree \* pop(){      tree \*t=NULL;      if(top!=NULL){          t=top->r;          top=top->next;      }      return t;  }  void inorder(tree \*r){        if(r!=NULL){          inorder(r->left);          printf("\t%d",r->data);          inorder(r->right);      }  }  void inorder\_wur(tree \*root){      tree \*t;      for(t=root;t!=NULL;t=t->left){          push(t);      }      t=pop();      while(t!=NULL){          printf("\t%d",t->data);          for(t=t->right;t!=NULL;t=t->left){              push(t);          }          t=pop();      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'12\_2\_bst\_inorder\_wur.exe'  BST OPERATIONS ->.  1.INSERT.  2.INORDER(Using reccusrion).  3.INORDER(Without using reccusrion).  4.EXIT  Enter any choice :1  Enter how many elements to insert(n) : 12  Enter the element to insert : 200  Enter the element to insert : 250  Enter the element to insert : 10  Enter the element to insert : 5  Enter the element to insert : 100  Enter the element to insert : 220  Enter the element to insert : 240  Enter the element to insert : 235  Enter the element to insert : 245  Enter the element to insert : 150  Enter the element to insert : 120  Enter the element to insert : 170  BST OPERATIONS ->.  1.INSERT.  2.INORDER(Using reccusrion).  3.INORDER(Without using reccusrion).  4.EXIT  Enter any choice :3  inorder Without Using Reccursion of entered BST is : 5 10 100 120 150 170 200 220 235 240 245 250  BST OPERATIONS ->.  1.INSERT.  2.INORDER(Using reccusrion).  3.INORDER(Without using reccusrion).  4.EXIT  Enter any choice :2  Inorder of entered BST is : 5 10 100 120 150 170 200 220 235 240 245 250  BST OPERATIONS ->.  1.INSERT.  2.INORDER(Using reccusrion).  3.INORDER(Without using reccusrion).  4.EXIT  Enter any choice :4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 12\_3** | **Date: 29/09/2022** |
| **Binary search tree insertion and display pre-order without using recursion.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 12\_3 : Binary search tree insertion and display pre-order without using recursion  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*left;      struct node \*right;  };  typedef struct node tree;  struct stack{      tree \*r;      struct stack \*next;  };  typedef struct stack stack;  tree \*root=NULL;  stack \*top=NULL;  void insert();  void push();  tree \* pop();  void preorder();  void preorder\_wur();  int menu();  int main(){      int ch,e,n,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  for(i=0;i<n;i++){                       printf("\nEnter the element to insert : ");                       scanf("%d",&e);                       insert(e);                  }                  break;              case 2:                  printf("\n\npreorder  of entered BST is : ");                  preorder(root);                  break;              case 3:                  printf("\n\npreorder Without Using Reccursion of entered BST is : ");                  preorder\_wur(root);                  break;              case 4:                  printf("\nExiting....");                  break;              default:                  printf("\nInvalid Input . Try again.\n");          }      }        return 0;  }  int menu(){      int ch;      printf("\nBST OPERATIONS ->.\n1.INSERT.\n2.PREORDER(Using reccusrion).\n3.PREORDER(Without using reccusrion).\n4.EXIT");      printf("\nEnter any choice :");      scanf("%d",&ch);      return ch;  }  void insert(int e){      tree \*p,\*x;      if(root==NULL){          root=(tree \*)malloc(sizeof(tree));          root->data=e;          root->left=NULL;          root->right=NULL;      }      else{          p=root;          while(p!=NULL){              x=p;              if(e < p->data){                  p=p->left;              }              else{                  p=p->right;              }          }          if(e > x->data){              x->right=(tree \*)malloc(sizeof(tree));              x->right->data=e;              x->right->left=NULL;              x->right->right=NULL;          }          else{              x->left=(tree \*)malloc(sizeof(tree));              x->left->data=e;              x->left->left=NULL;              x->left->right=NULL;          }      }  }  void push(tree \*t){      stack \*temp=(stack \*)malloc(sizeof(stack));      temp->r=t;      temp->next=top;      top=temp;  }  tree \* pop(){      tree \*t=NULL;      if(top!=NULL){          t=top->r;          top=top->next;      }      return t;  }  void preorder(tree \*r){        if(r!=NULL){          printf("\t%d",r->data);          preorder(r->left);          preorder(r->right);      }  }  void preorder\_wur(tree \*root){      tree \*t;      for(t=root;t!=NULL;t=t->left){          printf("\t%d",t->data);          push(t);      }      t=pop();      while(t!=NULL){          for(t=t->right;t!=NULL;t=t->left){              printf("\t%d",t->data);              push(t);          }          t=pop();      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'12\_3\_bst\_preorder\_wur.exe'  BST OPERATIONS ->.  1.INSERT.  2.PREORDER(Using reccusrion).  3.PREORDER(Without using reccusrion).  4.EXIT  Enter any choice :1  Enter how many elements to insert(n) : 12  Enter the element to insert : 200  Enter the element to insert : 10  Enter the element to insert : 250  Enter the element to insert : 220  Enter the element to insert : 240  Enter the element to insert : 235  Enter the element to insert : 245  Enter the element to insert : 5  Enter the element to insert : 100  Enter the element to insert : 150  Enter the element to insert : 120  Enter the element to insert : 170  BST OPERATIONS ->.  1.INSERT.  2.PREORDER(Using reccusrion).  3.PREORDER(Without using reccusrion).  4.EXIT  Enter any choice :3  preorder Without Using Reccursion of entered BST is : 200 10 5 100 150 120 170 250 220 240 235 245  BST OPERATIONS ->.  1.INSERT.  2.PREORDER(Using reccusrion).  3.PREORDER(Without using reccusrion).  4.EXIT  Enter any choice :2  preorder of entered BST is : 200 10 5 100 150 120 170 250 220 240 235 245  BST OPERATIONS ->.  1.INSERT.  2.PREORDER(Using reccusrion).  3.PREORDER(Without using reccusrion).  4.EXIT  Enter any choice :4 |

|  |  |
| --- | --- |
| **Program # 12\_4** | **Date: 29/09/2022** |
| **Binary search tree insertion and display post-order without using recursion.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 12\_4 : Binary search tree insertion and display post-order without using recursion  #include<stdio.h>  #include<malloc.h>  struct node{      int data;      struct node \*left;      struct node \*right;  };  typedef struct node tree;  struct stack{      tree \*r;      int count;      struct stack \*next;  };  typedef struct stack stack;  tree \*root=NULL;  stack \*top=NULL;  void insert();  void push();  tree \* pop();  void postorder();  void postorder\_wur();  int menu();  int main(){      int ch,e,n,i;      for(ch=menu();ch!=4;ch=menu()){          switch(ch){              case 1:                  printf("\nEnter how many elements to insert(n) : ");                  scanf("%d",&n);                  for(i=0;i<n;i++){                       printf("\nEnter the element to insert : ");                       scanf("%d",&e);                       insert(e);                  }                  break;              case 2:                  printf("\n\npostorder  of entered BST is : ");                  postorder(root);                  break;              case 3:                  printf("\n\npostorder Without Using Reccursion of entered BST is : ");                  postorder\_wur(root);                  break;              case 4:                  printf("\nExiting....");                  break;              default:                  printf("\nInvalid Input . Try again.\n");          }      }        return 0;  }  int menu(){      int ch;      printf("\nBST OPERATIONS ->.\n1.INSERT.\n2.POSTORDER(Using reccusrion).\n3.POSTORDER(Without using reccusrion).\n4.EXIT");      printf("\nEnter any choice :");      scanf("%d",&ch);      return ch;  }  void insert(int e){      tree \*p,\*x;      if(root==NULL){          root=(tree \*)malloc(sizeof(tree));          root->data=e;          root->left=NULL;          root->right=NULL;      }      else{          p=root;          while(p!=NULL){              x=p;              if(e < p->data){                  p=p->left;              }              else{                  p=p->right;              }          }          if(e > x->data){              x->right=(tree \*)malloc(sizeof(tree));              x->right->data=e;              x->right->left=NULL;              x->right->right=NULL;          }          else{              x->left=(tree \*)malloc(sizeof(tree));              x->left->data=e;              x->left->left=NULL;              x->left->right=NULL;          }      }  }  void push(tree \*t,int c){      stack \*temp=(stack \*)malloc(sizeof(stack));      temp->r=t;      temp->count=c;      temp->next=top;      top=temp;  }  tree \* pop(){      tree \*t=NULL;      if(top!=NULL){          t=top->r;          top=top->next;      }      return t;  }  int  peek()  {      int c = -1;      if(top != NULL)      {             c = top->count;      }      return c;  }  void postorder(tree \*r){        if(r!=NULL){          postorder(r->left);          postorder(r->right);          printf("\t%d",r->data);      }  }  void postorder\_wur(tree \*root){      tree \*t;      int c;      for(t=root;t!=NULL;t=t->left){          push(t,1);      }      c=peek();      t=pop();      while(t!=NULL){          if(c==1)          {              push(t,2);              for(t=t->right;t!=NULL;t=t->left){                    push(t,1);              }          }          else if (c==2)          {              printf("\t%d",t->data);          }          c=peek();          t=pop();      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'12\_4\_bst\_postorder\_wur.exe'  BST OPERATIONS ->.  1.INSERT.  2.POSTORDER(Using reccusrion).  3.POSTORDER(Without using reccusrion).  4.EXIT  Enter any choice :1  Enter how many elements to insert(n) : 12  Enter the element to insert : 200  Enter the element to insert : 250  Enter the element to insert : 10  Enter the element to insert : 220  Enter the element to insert : 240  Enter the element to insert : 235  Enter the element to insert : 245  Enter the element to insert : 5  Enter the element to insert : 100  Enter the element to insert : 150  Enter the element to insert : 120  Enter the element to insert : 170  BST OPERATIONS ->.  1.INSERT.  2.POSTORDER(Using reccusrion).  3.POSTORDER(Without using reccusrion).  4.EXIT  Enter any choice :3  postorder Without Using Reccursion of entered BST is : 5 120 170 150 100 10 235 245 240 220 250 200  BST OPERATIONS ->.  1.INSERT.  2.POSTORDER(Using reccusrion).  3.POSTORDER(Without using reccusrion).  4.EXIT  Enter any choice :2  postorder of entered BST is : 5 120 170 150 100 10 235 245 240 220 250 200  BST OPERATIONS ->.  1.INSERT.  2.POSTORDER(Using reccusrion).  3.POSTORDER(Without using reccusrion).  4.EXIT  Enter any choice :4  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 12\_5** | **Date: 29/09/2022** |
| **Binary search tree insertion using names and display the names in ascending order using inorder traversal.** | |

**Source Code:**

|  |
| --- |
| //PROGRAM 12\_5 : Binary search tree insertion using names and display the names in ascending order using inorder traversal.  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define SIZE 20  struct node  {      char data[SIZE];      struct node \*left, \*right;  };  typedef struct node tree;  tree \*root = NULL;  void insert(char e[SIZE])  {      tree \*t, \*p;      if (root == NULL)      {          root = (tree \*)malloc(sizeof(tree));          strcpy(root->data, e);          root->left = NULL;          root->right = NULL;      }      else      {          t = root;          while (t != NULL)          {              p = t;              if (strcmp(t->data, e) == 0)              {                  printf("Duplicate Key");                  return;              }              else if (strcmp(t->data, e) > 0)              {                  t = t->left;              }              else              {                  t = t->right;              }          }          if (strcmp(p->data, e) > 0)          {              p->left = (tree \*)malloc(sizeof(tree));              strcpy(p->left->data, e);              p->left->left = NULL;              p->left->right = NULL;          }          else          {              p->right = (tree \*)malloc(sizeof(tree));              strcpy(p->right->data, e);              p->right->left = NULL;              p->right->right = NULL;          }      }  }  void inorder(tree \*r)  {      if (r == NULL)          return;      inorder(r->left);      printf("\t%s", r->data);      inorder(r->right);  }  int main()  {      int ch,i,n;      char a[50];      while (1)      {          printf("\nBST operations using names : \n1.Insert\n2.Inorder\n3.Exit.");          printf("\nEnter your choice: ");          scanf("%d", &ch);          switch (ch)          {          case 1:              printf("\nEnter how many Names to insert(n) : ");                  scanf("%d",&n);                  printf("\nEnter %d names -> \n",n);                  for(i=0;i<n;i++){                       printf("-> ");                       scanf("%s", a);                       insert(a);                  }              break;          case 2:              printf("\nInorder of entered names are : \n");              inorder(root);              break;          case 3:              printf("\nExiting!!!");              exit(0);              break;          default:              printf("\nInvalid Choice.Try again");          }      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'12\_5\_bst\_using\_names.exe'  BST operations using names :  1.Insert  2.Inorder  3.Exit.  Enter your choice: 1  Enter how many Names to insert(n) : 8  Enter 8 names ->  -> Hari  -> Anshad  -> Abey  -> Binoy  -> Sreekumar  -> Nihal  -> Thomas  -> Jibin  BST operations using names :  1.Insert  2.Inorder  3.Exit.  Enter your choice: 2  Inorder of entered names are :  Abey Anshad Binoy Hari Jibin Nihal Sreekumar Thomas  BST operations using names :  1.Insert  2.Inorder  3.Exit.  Enter your choice: 3  Exiting!!!  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 13\_1** | **Date: 02/10/2022** |
| **Demonstrate the data structure of adjacent matrix using arrays.** | |

**Source Code:**

|  |
| --- |
| // PROGRAM 13\_1 : Demonstrate the data structure of adjacent matrix  using arrays.  #include <stdio.h>  #include <conio.h>  #define max 20  int adj[max][max];  int n;  void create\_graph();  void display();  int main()  {      create\_graph();      display();      return 0;  }  void create\_graph()  {      int i, max\_edges, origin, destin;      printf("\nEnter number of Vertices(nodes) : ");      scanf("%d", &n);      printf("\n[NOTE: Enter 0 0 to quit]");      max\_edges = n \* (n - 1);      for (i = 1; i <= max\_edges; i++)      {          printf("\nEnter edge %d  : \n", i);          scanf("%d %d", &origin, &destin);          if ((origin == 0) && (destin == 0))              break;          if (origin > n || destin > n || origin <= 0 || destin <= 0)          {              printf("Invalid edge!.Enter valid edge.\n");              i--;          }          else              adj[origin][destin] = 1;      }  }  void display()  {      int i, j;      printf("\n---- Adjacency matrix----\n");      for (i = 1; i <= n; i++)      {          for (j = 1; j <= n; j++)              printf("%4d", adj[i][j]);          printf("\n");      }  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'13\_1\_adjacent\_matrix\_using\_array.exe'  Enter number of Vertices(nodes) : 4  [NOTE: Enter 0 0 to quit]  Enter edge 1 :  1  2  Enter edge 2 :  1  4  Enter edge 3 :  2  1  Enter edge 4 :  2  4  Enter edge 5 :  2  3  Enter edge 6 :  3  4  Enter edge 7 :  3  2  Enter edge 8 :  4  1  Enter edge 9 :  4  2  Enter edge 10 :  4  3  Enter edge 11 :  0  0  ---- Adjacency matrix----  0 1 0 1  1 0 1 1  0 1 0 1  1 1 1 0  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |

|  |  |
| --- | --- |
| **Program # 13\_2** | **Date: 04/10/2022** |
| **Demonstrate the data structure of adjacent matrix using arrays.** | |

**Source Code:**

|  |
| --- |
| // PROGRAM 13\_2 : Demonstrate the data structure of adjacent matrix  using arrays.  #include <stdio.h>  #include <stdlib.h>  struct node {    int vertex;    struct node\* next;  };  struct node\* createNode(int);  struct Graph {    int numVertices;    struct node\*\* adjLists;  };  // Create a node  struct node\* createNode(int v) {    struct node\* newNode = malloc(sizeof(struct node));    newNode->vertex = v;    newNode->next = NULL;    return newNode;  }  // Create a graph  struct Graph\* createAGraph(int vertices) {    struct Graph\* graph = malloc(sizeof(struct Graph));    graph->numVertices = vertices;    graph->adjLists = malloc(vertices \* sizeof(struct node\*));    int i;    for (i = 0; i < vertices; i++)      graph->adjLists[i] = NULL;    return graph;  }  // Add edge  void addEdge(struct Graph\* graph, int s, int d) {    // Add edge from s to d    struct node\* newNode = createNode(d);    newNode->next = graph->adjLists[s];    graph->adjLists[s] = newNode;    // Add edge from d to s    newNode = createNode(s);    newNode->next = graph->adjLists[d];    graph->adjLists[d] = newNode;  }  // Print the graph  void printGraph(struct Graph\* graph) {    int v;    for (v = 0; v < graph->numVertices; v++) {      struct node\* temp = graph->adjLists[v];      printf("\n Vertex %d\n: ", v);      while (temp) {        printf("%d -> ", temp->vertex);        temp = temp->next;      }      printf("\n");    }  }  int main() {    struct Graph\* graph = createAGraph(4);    printf("\nRepresentation of adjacency List --->\n");    addEdge(graph, 0, 1);    addEdge(graph, 0, 2);    addEdge(graph, 0, 3);    addEdge(graph, 1, 2);    printGraph(graph);    return 0;  } |

**OUTPUT:**

|  |
| --- |
| PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> & .\'13\_2\_adjacent\_matrix\_using\_linkedlist.exe'  Representation of adjacency List --->  Vertex 0  : 3 -> 2 -> 1 ->  Vertex 1  : 2 -> 0 ->  Vertex 2  : 1 -> 0 ->  Vertex 3  : 0 ->  PS E:\MCA\LAB \_DS\C LAB SESSIONS\DONE\output> |