**Data Structures Lab**

**(JOURNAL)**

**NAME: SHARVARI BIRAJDAR**

**ROLL NO: 1409**

**CLASS: FY MCA (2023-25)**

# Data Structures Lab – Batch B1

|  |  |  |
| --- | --- | --- |
| ***Unit 1*** | ***Sorting*** | ***Date*** |
| 1. | Bubble Sort | 01.09.2023 |
| 2. | Selection Sort | 04.09.2023 |
| 3. | Insertion Sort | 09.09.2023 |
| 4. | Radix Sort | 11.09.2023 |
| 5. | Shell Sort | 16.09.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 2*** | ***Searching*** | ***Date*** |
| 6. | Linear Search | 25.08.2023 |
| 7. | Binary Search | 28.08.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 2*** | ***Hashing*** | ***Date*** |
| 8. | Modulo Division with Linear Probe | 11.12.2023 |
| 9. | Digit Extraction with Linear Probe | 25.12.2023 |
| 10. | Fold Shift with Linear Probe | 25.12.2023 |
| 11. | Fold Boundary with Linear Probe | 25.12.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 3*** | ***Stacks*** | ***Date*** |
| 12. | Array Based Stack | 30.10.2023 |
| 13. | List Based Stack | 04.11.2023 |
| 14. | Evaluation of Postfix Expression using Stacks | 06.11.2023 |
| 15. | Balancing of Parenthesis using Stacks | 09.11.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 4*** | ***Queues*** | ***Date*** |
| 16. | Ordinary Queue | 09.11.2023 |
| 17. | Circular Queue | 25.11.2023 |
| 18. | Priority Queue | 04.12.2023 |
| 19. | Double-Ended Queue | 02.12.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 5*** | ***Linked Lists*** | ***Date*** |
| 20. | Singly Linked List | 18.09.2023  30.09.2023 |
| 21. | Circular Linked List | 09.10.2023 |
| 22. | Doubly Linked List | 14.10.2023  16.10.2023 |
| 23. | Polynomial Addition using Lists | 21.10.2023  28.10.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 6*** | ***Trees*** | ***Date*** |
| 24. | Binary Search Trees | 05.12.2023 |
| 25. | Max-Heaps | 11.12.2023 |
| 26. | Min-Heaps | 25.12.2023 |

|  |  |  |
| --- | --- | --- |
| ***Unit 7*** | ***Graphs*** | ***Date*** |
| 27. | Representation of Graphs using Adjacency Matrix | 07.12.2023 |
| 28. | Perform Breadth-First Traversal on a Graph using Queues | 07.12.2023 |
| 29. | Find MST using Kruskal’s Algorithm | 09.12.2023 |

## Unit-1 : Sorting

|  |  |
| --- | --- |
| **Program No:** | **1.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Bubble Sort** |
| **Objective :** | **Unit-1: Sorting** |

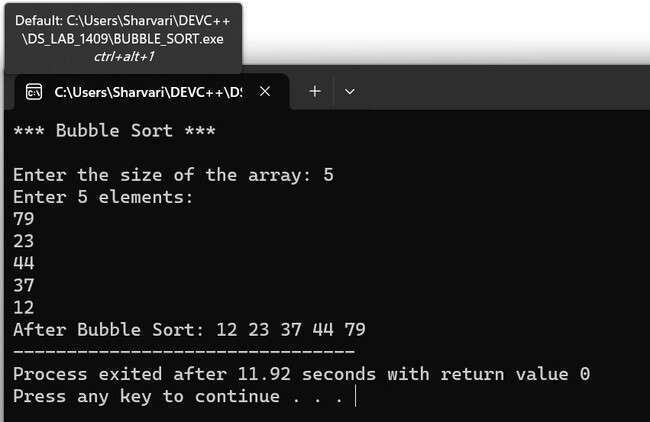
**Date:** 01.09.2023

**Source Code:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* | Name:  Roll No: 1409    #    using    int main  { | Sharvari Birajdar    Unit 1: Sorting  Program: Bubble Sort\*/ include<iostream> namespace std;  ()  //Variables int A[20], n;  int i,j,temp=0;    cout << "\*\*\* Bubble Sort \*\*\*" << endl << endl;    cout << "Enter the size of the array: "; cin >> n;    cout <<"Enter " << n << " elements: " << endl; for(i=0; i<n; i++)  {  cin >> A[i];  }    //Bubble Sort for(i=0 ; i<n-1; i++)  {  for(j=i+1; j<n; j++) |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | {  if(A[i] > A[j])  {  temp = A[i]; A[i] = A[j];  A[j] = temp;  }  }//end of for j  }//end of for i    //Display the sorted array cout << "After Bubble Sort: ";  for(i=0 ; i<n; i++)  {  cout << A[i] << " ";  }    }//end of main | | |

**Output**

**:**



|  |  |
| --- | --- |
| **Program No:** | **2.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Selection Sort** |
| **Objective :** | **Unit-1: Sorting** |

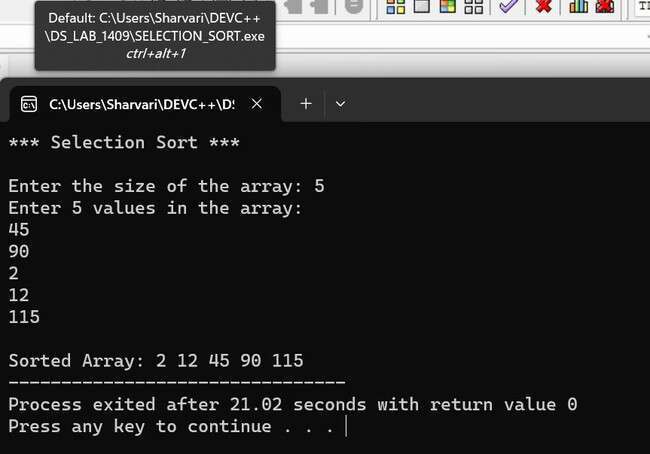
**Date:** 04.09.2023

**Source Code:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* | Roll No: 1409    #    **using**    **int** **main**  { | Name: Sharvari Birajdar    Unit 1: Sorting  Program: Selection Sort\*/ include<iostream> **namespace** std;  ()  //Variables  **int** a[**20**], n,temp,loc; **int** i,j;  **int** min;    cout << "\*\*\* Selection Sort \*\*\*" << endl << endl;    cout << "Enter the size of the array: "; cin >> n;    cout << "Enter " << n <<" values in the array: " << endl; **for**(i=**0**; i<n; i++)  {  cin >> a[i];  }    //Selection Sort **for**(i=**0**; i<n; i++)  {  min = a[i]; // assume min value is first value in array loc = i; // and its location is i |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | **for**(j=i+**1**; j<n; j++) // check through array for a smaller  { // value than min **if**(a[j] < min)  {  min = a[j];  loc = j;  }  }//end of inner for j    //For this iteration, you have identified the min and its loc  temp = a[i]; a[i] = a[loc]; a[loc] = temp;  }//end for i    cout << endl << "Sorted Array: ";  **for**(i=**0**; i<n; i++)  {  cout << a[i] << " ";  }  }//end of main | | |

**Output**

**:**



|  |  |
| --- | --- |
| **Program No:** | **3.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Insertion Sort** |
| **Objective :** | **Unit-1: Sorting** |

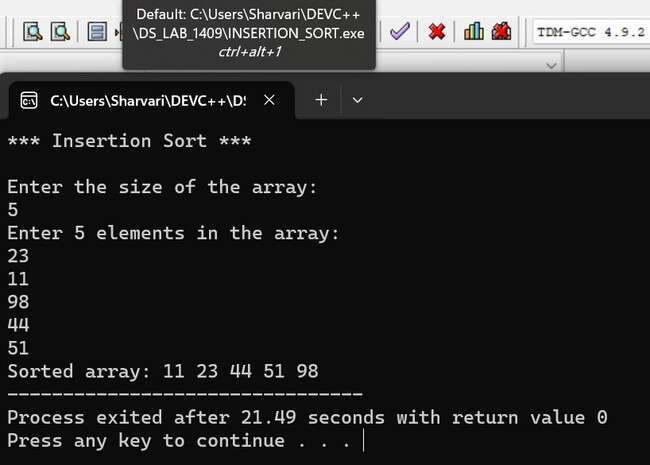
**Date:** 09.09.2023

**Source Code:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* | Roll No: 1409    # **using**    **int** **main**  { | Name: Sharvari Birajdar    Unit 1: Sorting  Program: Insertion Sort\*/  include<iostream> **namespace** std;  ()  //Variables **int** a[**20**],n;  **int** i,temp,loc;    cout << "\*\*\* Insertion Sort \*\*\*"<< endl << endl;    cout << "Enter the size of the array: " << endl; cin >> n;    cout << "Enter " <<n<<" elements in the array: " << endl; **for**(i=**0**;i<n;i++)  {  cin>>a[i];  }    //Insertion sort  //i=1 as the first element is assumed to be sorted. **for**(i=**1**;i<n;i++)  {  temp=a[i]; //first element of unsorted region loc=i-**1**; //location to the left of unsorted region  //Start with assuming that the first element is sorted already |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | **while**(temp<a[loc] && loc>=**0**)  {  a[loc+**1**]=a[loc];  loc--;  }//end of while    a[loc+**1**] = temp; //Write temp to its final position    }//end of for i      cout << "Sorted array: ";  **for**(i=**0**;i<n;i++)  {  cout << a[i] << " ";  }  }//end of main | | |

**Output**

**:**



|  |  |
| --- | --- |
| **Program No:** | **4.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Radix Sort** |
| **Objective :** | **Unit-1: Sorting** |

**Date:** 11.09.2023

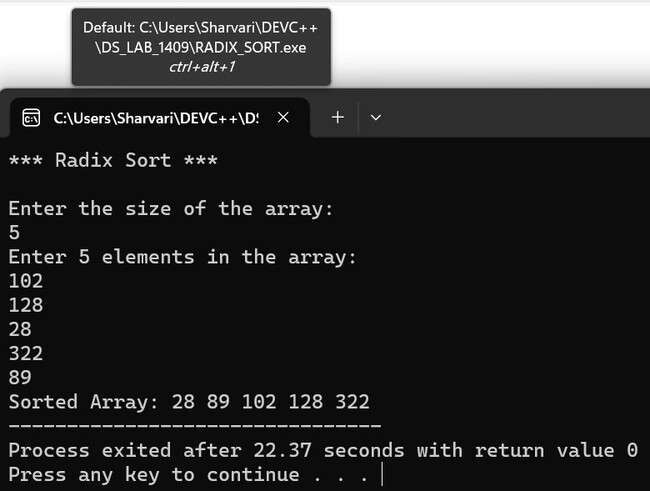
**Source Code:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* | # **using**    **int** { | Name: Sharvari Birajdar  Roll No: 1409  Unit 1: Sorting  Program: Radix Sort\*/  include<iostream> **namespace** std; **main**()  //Variables **int** a[**20**], n, i,j; **int** max,r; **int** passes=**0**,pass\_no,divisor=**1**;  **int** k,bucket[**10**][**20**], b\_count[**10**];    cout << "\*\*\* Radix Sort \*\*\***\n\n**";    cout << "Enter the size of the array: " << endl; cin >> n;    cout << "Enter " << n << " elements in the array: " << endl; **for**(**int** i=**0**;i<n;i++)  {  cin>>a[i];  }    //Radix Sort    //1. Identify the maximum number in the array max = a[**0**]; **for**(i=**1**; i<n; i++) |

|  |  |
| --- | --- |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | {  **if**(a[i] > max)  {  max=a[i];  }  }    //2. Count digits in max  **while**(max>**0**)  {  passes++;  max = max/**10**;  }    //3. Radix Sort Process  **for**(pass\_no=**1**; pass\_no<=passes; pass\_no++)  {  **for**(k=**0**; k<**10**; k++)//10 for 0-9 bins  {  b\_count[k] = **0**; //Initialize b\_count to 0 to indicate elements in the row,set to all 0 i.e to set empty bucket  }    **for**(i=**0**; i<n; i++) // Assign elements to the bin  {  //r-units position  //bucket- to track counter, avoid overwriting    r=(a[i]/divisor)%**10**;  bucket[r][b\_count[r]] = a[i];  b\_count[r]++;  }    //collecting elements from the bucket or bins i=**0**;  **for**(k=**0**; k<**10**;k++)  {  **for**(j=**0**; j<b\_count[k]; j++)  {  a[i] = bucket[k][j];  i++; |
| 74  75  76  77  78  79  80  81  82  83  84  85  86 | }//inner for j  }//outer for k    divisor=divisor\***10**; // to make it work for every digit    }//end of for pass\_no    cout << "Sorted Array: ";  **for**(i=**0**; i<n;i++)  {  cout << a[i] << " ";  }    }//end of main |

**Output**

**:**



|  |  |
| --- | --- |
| **Program No:** | **5.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Shell Sort** |
| **Objective :** | **Unit-1: Sorting** |

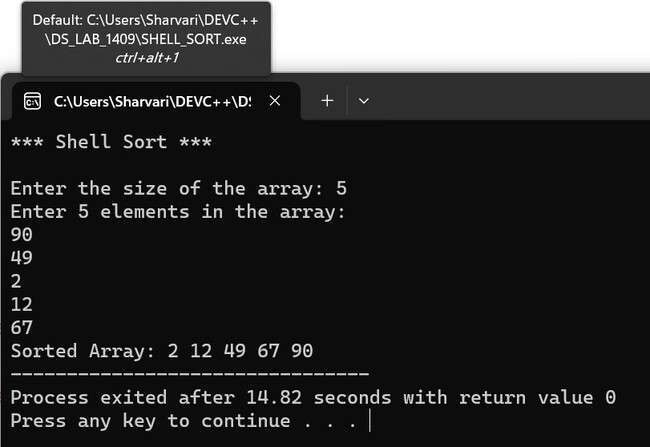
**Date:** 16.09.2023

**Source Code:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* | #  #    **using**    **int** { | Name: Sharvari Birajdar  Roll No: 1409  Unit 1: Sorting  Program: Shell Sort\*/  include<iostream> include<math.h> **namespace** std; **main**()  **int** A[**20**],n,i;  **int** gap,pos,extractItem;     |  |  | | --- | --- | | "\*\*\* Shell Sort \*\*\*" | << | | "Enter the size of the array: " | |   cout << endl << endl;  cout << ; cin >> n;    cout << "Enter " << n << " elements in the array:**\n**"; **for**(i=**0**; i<n; i++)  {  cin >> A[i];  }    //Shell Sort gap = floor(n/**2**);  **while**(gap>**0**)  {  **for**(i=**0**; i<n-gap; i++)  {  extractItem = A[i+gap]; |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | pos = i+gap;    **while**((pos-gap)>=**0** && extractItem<A[pos-gap])  {  A[pos] = A[pos-gap]; pos = pos-gap;  }//end of inner while    A[pos] = extractItem;  }//end of i  gap = floor(gap/**2**);  }//end of while gap    cout << "Sorted Array: ";  **for**(i=**0**; i<n; i++)  {  cout << A[i] << " ";  }  }//end of main | | |

**Output**

**:**



## Unit - 2 : Searching

|  |  |
| --- | --- |
| **Program No:** | **6.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Linear Search** |
| **Objective :** | **Unit-2: Searching** |

**Date:** 25.08.2023

**Source Code:**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | /\*            **using**    **int** { | Name: Sharvari Birajdar  Roll No: 1409  Unit: 2  Program: Linear Search \*/  #include<iostream> #include<conio.h> **namespace** std; **main**()  **int** num[**10**]; **int** target; **int** size, i;  **int** flag=**0**; // Flag down - target not found    cout << "\*\*\* Linear Search \*\*\*" << endl << endl;    cout << "Enter the size of the array: "; cin >> size;    cout << "Enter " << size << " values in the array: " << endl; **for**(i=**0** ; i<size ; i++)  {  cin >> num[i];  }    cout << "Enter the target value: "; |

|  |  |
| --- | --- |
| 30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | cin >> target;    //Linear Search  **for**(i=**0** ; i<size ; i++)  {  **if**(num[i] == target)  {  flag = **1**; //target is found - raise the flag  **break**;  }  }    **if**(flag == **0**)  {  cout << target << " not found!";  }  **else**  {  cout << target << " found at index " << i;  }  }//end of main |

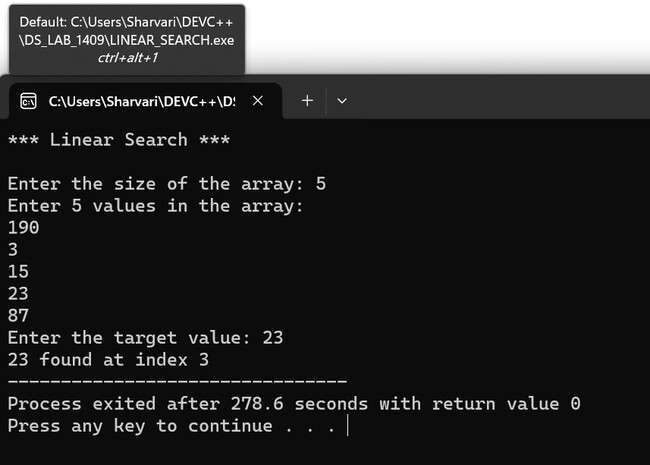
**Output**

**:**

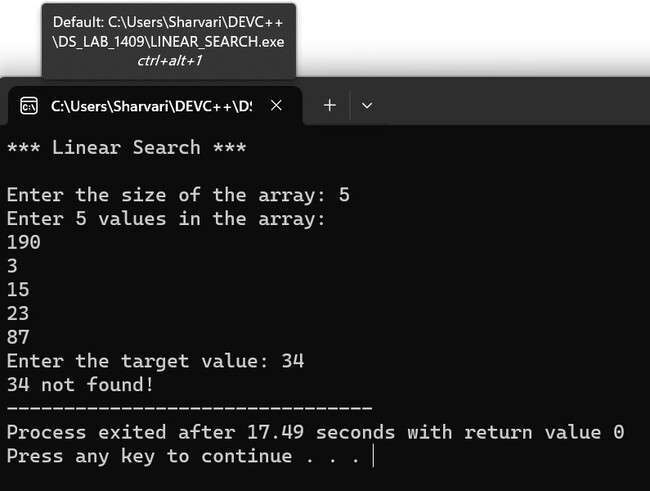
(

Target Found

)



(Target Not Found)



|  |  |
| --- | --- |
| **Program No:** | **7.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Binary Search** |
| **Objective :** | **Unit-2: Searching** |

**Date:** 28.08.2023

**Source Code:**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\*            **using**    **int** **main**  { | Name: Sharvari Birajdar  Roll No: 1409  Unit 2: Searching  Program: Binary Search \*/    #include<iostream> **namespace** std;  ()  //variables  **int** num[**10**], size, target, first, last, mid, i, flag=**0**;    cout << "\*\*\* Binary Search \*\*\*" << endl << endl;    cout << "Enter the size of the array: "; cin >> size;    cout << "Enter " << size << " elements in sorted order:" << endl;    //reading elements **for**(i=**0** ; i <size ; i++)  {  cin >> num[i];  }    cout << "Enter the target value: "; cin >> target;    //Binary Search first = **0**; |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | last = size-**1**;    **while**(first<=last)  {  mid = (first+last)/**2**;    **if**(target == num[mid])  {  flag = **1**; //target is found - raise the flag  **break**;  }    **if**(target < num[mid])  {  last = mid-**1**;  }  **else** //control reaches here only if target num[mid]  {  first = mid+**1**;  }  }//end of while    **if**(flag == **0**)  {  cout << target << " is not found";  }  **else**  {  cout << target << " is found at index " << mid;  }    }//end of main | |

**Output:**

(

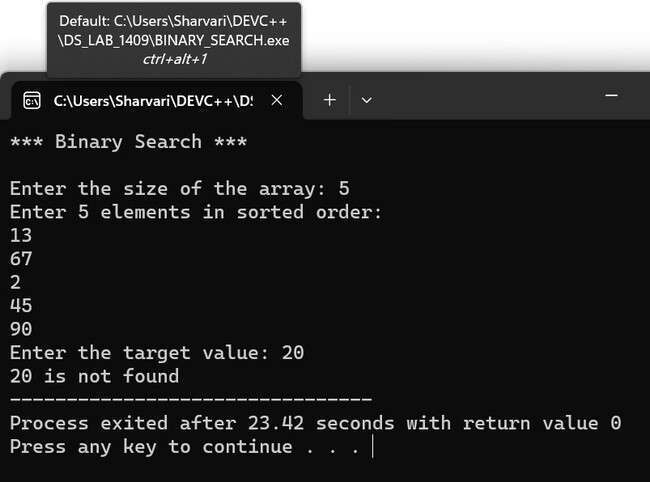
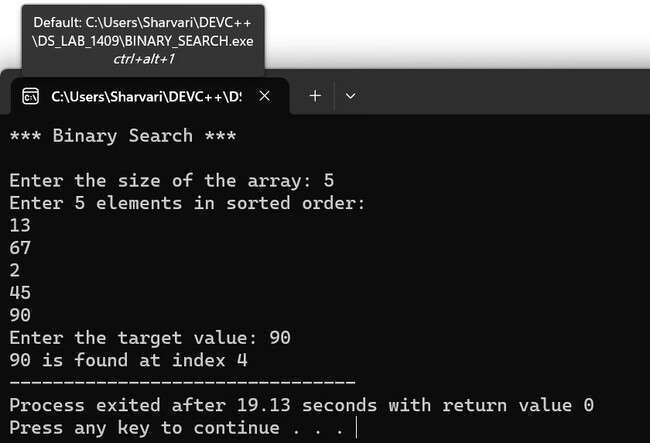
Target Found

)

(

Target Not Found

)



## Unit - 2 : Hashing

|  |  |
| --- | --- |
| **Program No:** | **8.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Modulo Division with Linear Probe** |
| **Objective :** | **Unit-2: Hashing** |

**Date:** 11.12.2023 **Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | */\* Name: Sharvari Birajdar*  *Roll no: 1409*  *Unit 2: Hashing*  *Program: Modulo division hashing with linear probe for collision\*/*    #include<iostream>  #include<conio.h>    #define MAX 10  #define EMPTY -1    **using** **namespace** std;    */\*----------1. Node Template -Not required----------\*/*    */\*----------2.Array Based Template for Hash Table--------\*/* **class** **Hash**  { int h[MAX];    public: Hash()  { int i;  **for**(i=0; i<MAX; i++)  {  h[i] = EMPTY;  }  } |

|  |  |
| --- | --- |
| 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | void Insert(int key); void Display(); void Search(int skey);  };    */\*----------------3.Functions------------------\*/*    */\*----------------Insert------------------\*/*  void Hash::Insert(int key)  {  *//HASH FNC*  int addr = key % MAX; *//Modulo Division Hash Function*    int oaddr = addr; *///saving original address*    **if**(h[addr] == EMPTY ) *//addr is available*  {  h[addr] = key;  cout << key << " is inserted at address " << addr << endl;  } **else**  {  *//Collision*  **do**  {  addr = (addr+1) % MAX;  **if**(addr == oaddr)  {  cout << "Hash Table is Full!"; **return**;  }  }  **while**(h[addr]!=EMPTY);    h[addr] = key;  cout << key << " is inserted at address " << addr << endl;    }*//end of if-else*    }*//end of Insert* |

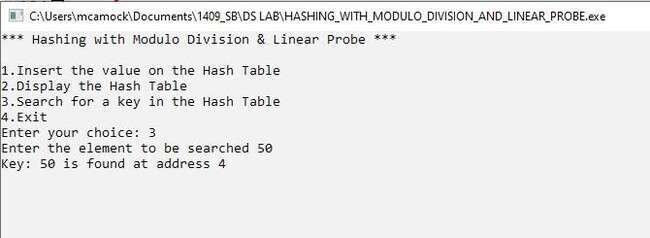
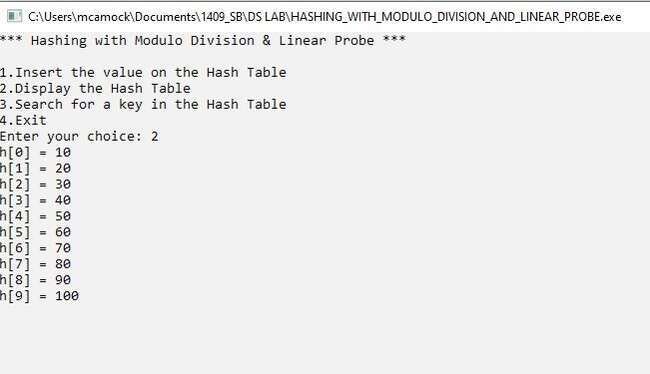
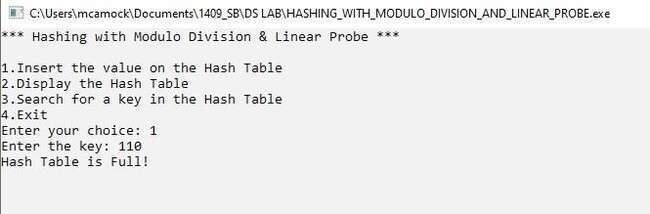
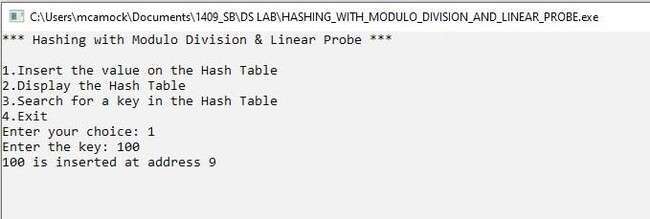
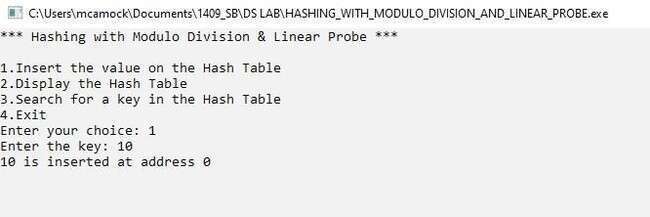
|  |  |
| --- | --- |
| 73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113 | */\*----------------Display------------------\*/*  void Hash::Display()  {  int i;    **for**(i=0;i<MAX;i++)  {  cout << "h[" << i << "] = ";  **if**(h[i] == EMPTY)  {  cout << "EMPTY" << endl;  } **else**  {  cout << h[i] << endl;  }  }  }*//end of Display*    */\*----------------Search------------------\*/*  void Hash::Search(int skey)  {  int saddr = skey%MAX; int oaddr = saddr;    **if**((h[saddr]) == skey)  {  cout << "Key: " << skey << " is found at address " << saddr << endl;  } **else** { **do**  {  saddr = (saddr+1) % MAX;  **if**(saddr == oaddr)  {  **break**;  }  }  **while**(h[saddr]!=skey); |

|  |  |
| --- | --- |
| 114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154 | **if**((h[saddr]) == skey)  {  cout << "Key: " << skey << " is found at address " << saddr << endl;  } **else**  {  cout << "Key: " << skey << " is not found" << endl;  }  }    }*//end of Search*    */\*------------------4.Menu--------------\*/* int main()  { int ch,num;  Hash h;    **while**(1)  {  system("cls");  cout << "\*\*\* Hashing with Modulo Division & Linear Probe \*\*\***\n\n**";    cout << "1.Insert the value on the Hash Table**\n**"; cout << "2.Display the Hash Table**\n**";  cout << "3.Search for a key in the Hash Table**\n**"; cout << "4.Exit**\n**";    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  {  **case** 1:  cout << "Enter the key: "; cin >> num;  h.Insert(num);  getch(); **break**; |

|  |  |
| --- | --- |
| 155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178 | **case** 2:  h.Display(); getch();  **break**;  **case** 3:  cout << "Enter the element to be searched "; cin >> num;  h.Search(num);  getch();  **break**;  **case** 4:  exit(1);  default:  cout << "Incorrect Choice!"; getch();  }*//end of switlch*    }*//end of while*    }*//end of main* |

**Output**

**:**



|  |  |
| --- | --- |
| **Program No:** | **9.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Digit Extraction with Linear Probe** |
| **Objective :** | **Unit-2: Hashing** |

**Date:** 25.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 2: Hashing  Program: Digit Extraction Hashing with linear probe for collision\*/    #include<iostream>  #include<conio.h>    #define MAX 4  #define EMPTY -1    **using** **namespace** std;    /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Hash Table--------\*/ **class** **Hash**  {  **int** h[MAX];    **public:** Hash()  { **int** i;  **for**(i=**0**; i<MAX; i++)  {  h[i] = EMPTY;  }  }    **void** Insert(**int** key); |

|  |  |
| --- | --- |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | **void** **Display**(); **void** **Search**(**int** skey);  **int** **DigitExtractionHash**(**int** key);  };    /\*----------------3.Functions------------------\*/    /\*----------------Insert------------------\*/  **void** Hash::Insert(**int** key)  {  //Digit Extraction Hash Function  **int** addr = DigitExtractionHash(key);    **int** oaddr = addr; //saving original address    **if**(h[addr] == EMPTY ) //addr is available  {  h[addr] = key;  cout << key << " is inserted at address " << addr << endl;  } **else**  {  //Collision  **do**  {  addr = (addr+**1**) % MAX;  **if**(addr == oaddr)  {  cout << "Hash Table is Full!"; **return**;  }  }  **while**(h[addr]!=EMPTY);    h[addr] = key;  cout << key << " is inserted at address " << addr << endl;    }//end of if-else    }//end of Insert |

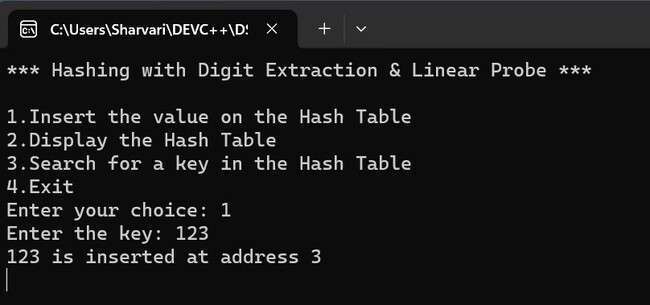
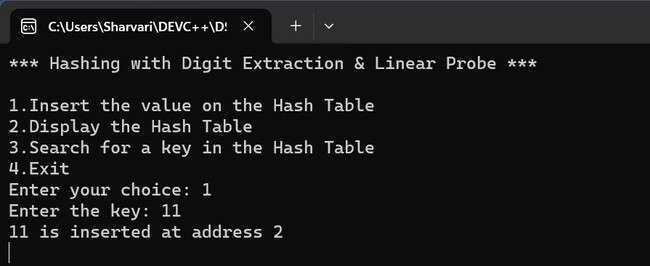
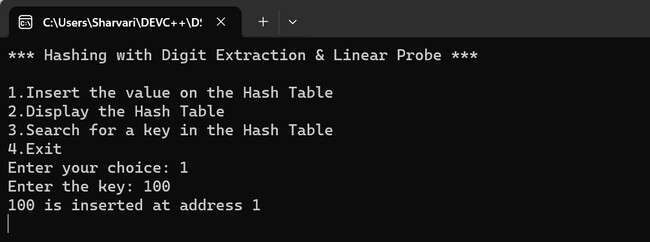
|  |  |
| --- | --- |
| 74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114 | /\*----------------Display------------------\*/  **void** Hash::Display()  {  **int** i;    **for**(i=**0**;i<MAX;i++)  {  cout << "h[" << i << "] = ";  **if**(h[i] == EMPTY)  {  cout << "EMPTY" << endl;  } **else**  {  cout << h[i] << endl;  }  }  }//end of Display    /\*----------------Search------------------\*/  **void** Hash::Search(**int** skey)  {  **int** saddr = DigitExtractionHash(skey);  **int** oaddr = saddr;    **if**((h[saddr]) == skey)  {  cout << "Key: " << skey << " is found at address " << saddr << endl;  } **else** { **do**  {  saddr = (saddr+**1**) % MAX;  **if**(saddr == oaddr)  {  **break**;  }  }  **while**(h[saddr]!=skey); |

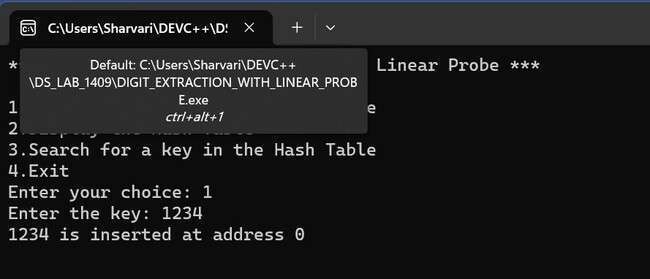
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155 | **if**((h[saddr]) == skey)  {  cout << "Key: " << skey << " is found at address " << saddr << endl;  } **else**  {  cout << "Key: " << skey << " is not found" << endl;  }  }    }//end of Search    /\*-----------DigitExtractionHash---------------\*/ **int** Hash::DigitExtractionHash(**int** key)  {  //Digit Extraction Hash Function: Sum of the digits  **int** sum=**0**;  **while**(key>**0**)  {  sum = sum + (key%**10**);  key = key/**10**;    }//end of while    **return** (sum % MAX);    }//end ofDigitExtractionHash    /\*------------------4.Menu--------------\*/ **int** main()  { **int** ch,num;  Hash h;    **while**(**1**)  { system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* Hashing with Digit Extraction & Linear Probe \*\*\***\n\n**" | |   cout << ; |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196 | |  |  |  | | --- | --- | --- | | "1.Insert the value on the Hash Table**\n**" | | | | "2.Display the Hash Table**\n**" | | ; | | "3.Search for a key in the Hash Table**\n**" | | | | "4.Exit**\n**" | ; | |   cout << ; cout <<  cout << ; cout <<    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  {  **case** **1**:  cout << "Enter the key: "; cin >> num;  h.Insert(num);  getch();  **break**;  **case** **2**:  h.Display(); getch();  **break**;  **case** **3**:  cout << "Enter the element to be searched "; cin >> num;  h.Search(num);  getch();  **break**;  **case** **4**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch();  }//end of switlch    }//end of while    }//end of main |

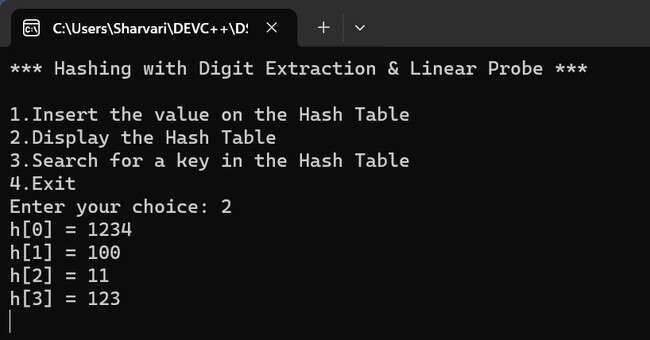
**Output:**

(Insertion)

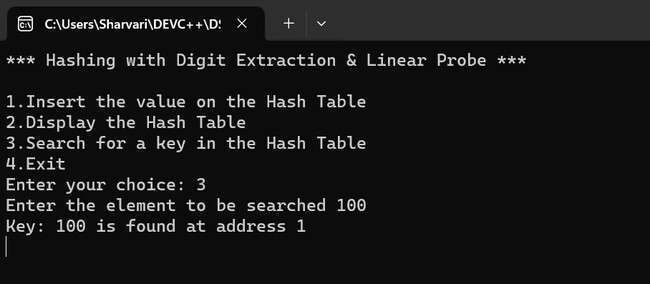




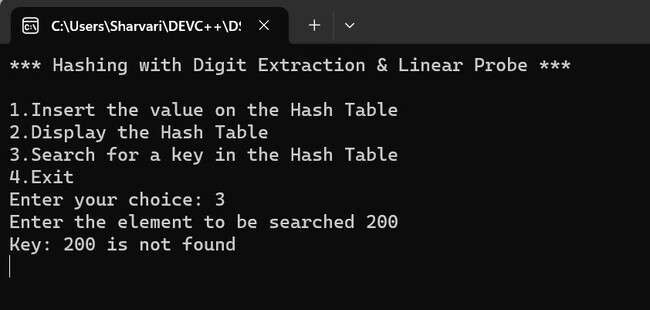
(Display)



(Target Found)



(Target Not Found)



|  |  |
| --- | --- |
| **Program No:** | **10.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Fold Shift with Linear Probe** |
| **Objective :** | **Unit-2: Hashing** |

**Date:** 25.12.2023 **Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 2: Hashing  Program: Fold Shift Hashing with linear probe for collision\*/    #include<iostream>  #include<conio.h>    #define MAX 19  #define EMPTY -1    **using** **namespace** std;    /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Hash Table--------\*/ **class** **Hash**  {  **int** h[MAX];    **public:**  Hash()  {  **for** (**int** i = **0**; i < MAX; i++)  {  h[i] = EMPTY;  }  }    **void** Insert(**int** key); **void** **Display**(); |

|  |  |
| --- | --- |
| 34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | **void** **Search**(**int** skey); **int** **FoldShiftHash**(**int** key);  };    /\*----------------3.Functions------------------\*/    /\*----------------Insert------------------\*/  **void** Hash::Insert(**int** key)  {  // Fold Shift Hash Function  **int** addr = FoldShiftHash(key);    **int** oaddr = addr; // saving original address    **if** (h[addr] == EMPTY)  {  h[addr] = key;  cout << key << " is inserted at address " << addr << endl;  } **else**  {  // Collision  **do**  {  addr = (addr + **1**) % MAX;  **if** (addr == oaddr)  {  cout << "Hash Table is Full!"; **return**;  }  } **while** (h[addr] != EMPTY);    h[addr] = key;  cout << key << " is inserted at address " << addr << endl; }//end of if-else    }//end of Insert    /\*----------------Display------------------\*/ **void** Hash::Display() |

|  |  |
| --- | --- |
| 75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115 | {  **int** i;    **for** (i = **0**; i < MAX; i++)  {  cout << "h[" << i << "] = "; **if** (h[i] == EMPTY)  {  cout << "EMPTY" << endl;  } **else**  {  cout << h[i] << endl;  }  }  }//end of Display    /\*----------------Search------------------\*/    **void** Hash::Search(**int** skey)  {  **int** saddr = FoldShiftHash(skey);  **int** oaddr = saddr;    **if** ((h[saddr]) == skey)  {  cout << "Key: " << skey << " is found at address " << saddr << endl;  } **else** { **do** {  saddr = (saddr + **1**) % MAX;    **if** (saddr == oaddr)  {  **break**;  }  } **while** (h[saddr] != skey);    **if** ((h[saddr]) == skey) |

|  |  |
| --- | --- |
| 116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156 | {  cout << "Key: " << skey << " is found at address " << saddr << endl;  } **else**  {  cout << "Key: " << skey << " is not found" << endl;  }  }  }//end of search    /\*------------FoldShiftHash---------------\*/    **int** Hash::FoldShiftHash(**int** key)  {  // Fold Shift Hash Function:  // Sum two digits at a time, discard MSB if three digits **int** sum = **0**; **int** tempKey = key;    **while** (tempKey > **0**)  {  // Extract the last two digits **int** twoDigits = (tempKey % **100**); sum += twoDigits;    // If sum becomes a three-digit number, discard the MSB **if** (sum >= **100**)  {  sum %= **100**;  }    tempKey /= **100**; // Move to the next two digits  }    **return** (sum % MAX);  }//end of FoldShiftHash    /\*------------------4.Menu--------------\*/ **int** main()  { |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197 | **int** ch, num;  Hash h;    **while** (**1**)  {  system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* Hashing with Fold Shift & Linear Probe \*\*\***\n\n**" | |   cout << ;     |  |  |  | | --- | --- | --- | | "1.Insert the value on the Hash Table**\n**" | | | | "2.Display the Hash Table**\n**" | | ; | | "3.Search for a key in the Hash Table**\n**" | | | | "4.Exit**\n**" | ; | |   cout << ; cout <<  cout << ; cout <<    cout << "Enter your choice: "; cin >> ch;    **switch** (ch)  {  **case** **1**:  cout << "Enter the key: "; cin >> num;  h.Insert(num); getch();  **break**;    **case** **2**:  h.Display(); getch();  **break**;    **case** **3**:  cout << "Enter the element to be searched: "; cin >> num;  h.Search(num); getch();  **break**;    **case** **4**: exit(**1**);    **default:** |

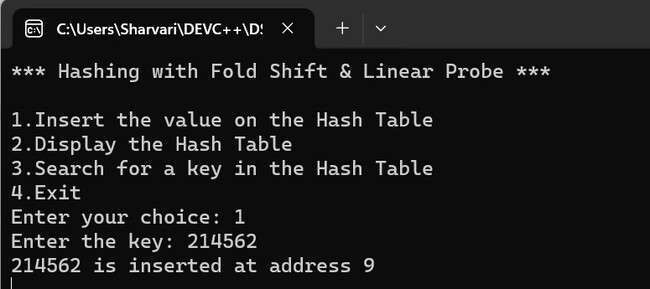
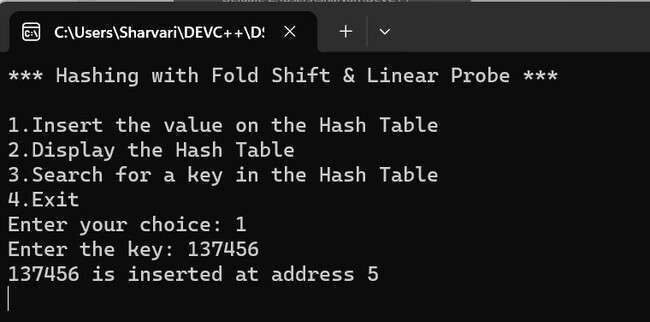
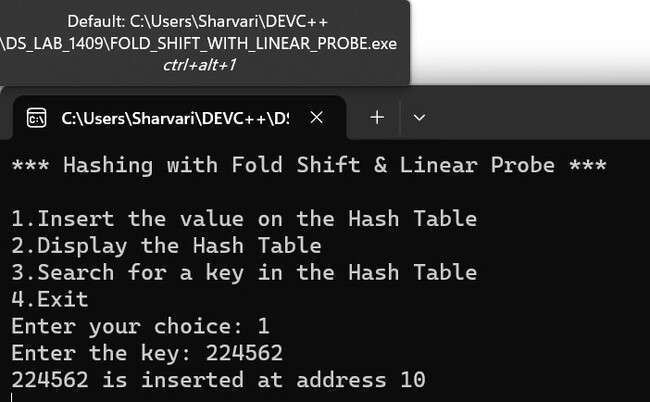
|  |  |
| --- | --- |
| 198  199  200  201  202  203  204  205  206  207 | cout << "Incorrect Choice!"; getch();  }//end of switch  }//end of while    **return** **0**; }//end of main |

**Output:**

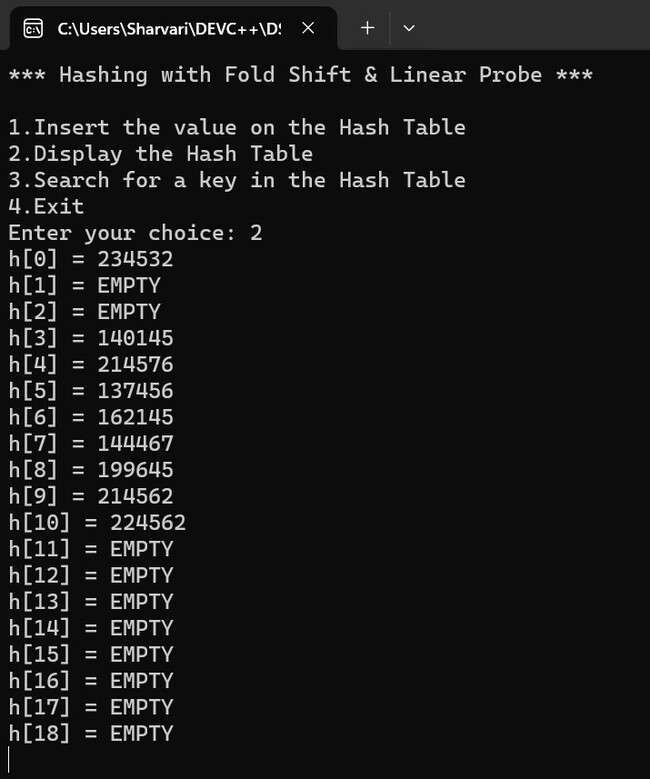
(

Insertion

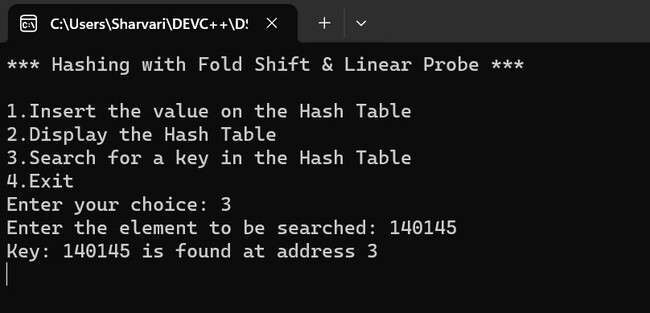
)



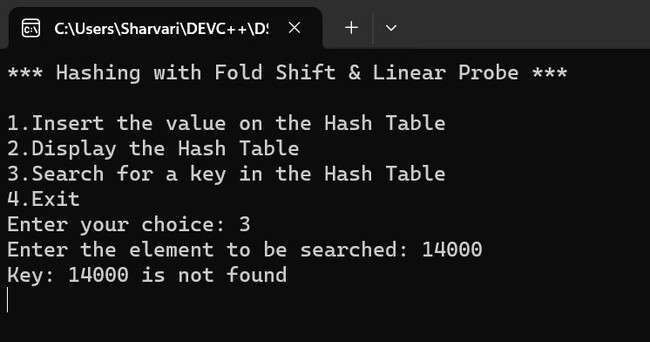
(Display)



(Target Found)



(Target Not Found)



|  |  |
| --- | --- |
| **Program No:** | **11.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Fold Boundary with Linear Probe** |
| **Objective :** | **Unit-2: Hashing** |

**Date:** 25.12.2023

**Source Code:**

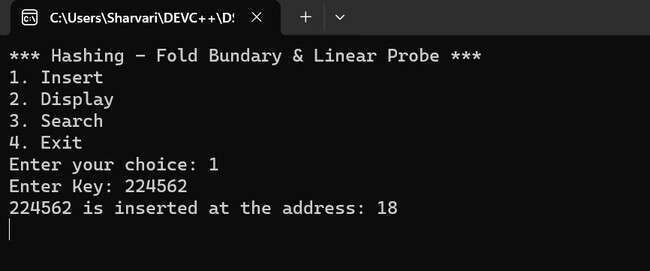
|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32** | **/\* Name: Sharvari Birajdar**  **Roll no: 1409**  **Unit 2: Hashing**  **Program: Fold Boundary Hashing with linear probe for collision\*/**    **#define MAX 19**  **#define EMPTY -1**    **#include<iostream>**  **#include<conio.h>**  **using namespace std;**    **//1. Node template - Not required**      **//2. Array based template for hash table**  **class Hash**  **{**  **int h[MAX];**    **public:**  **Hash()**  **{**  **for (int i=0; i<MAX; i++)**  **{**  **h[i] = EMPTY;**  **}**  **}**  **void Insert(int key);**  **void Display();**  **void Search(int key);**  **};** |

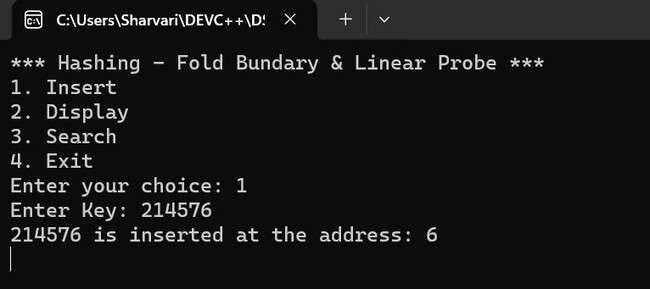
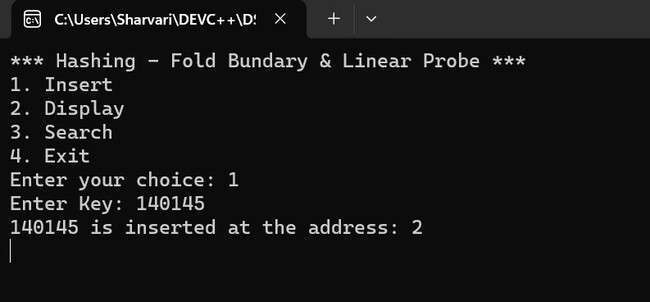
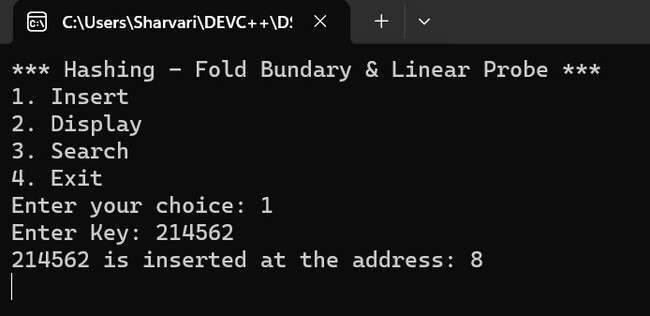
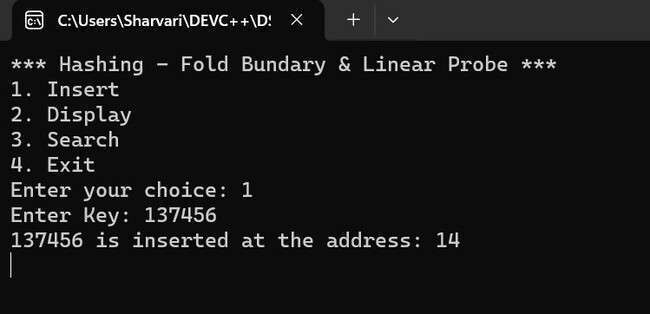
|  |  |
| --- | --- |
| **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53**  **54**  **55**  **56**  **57**  **58**  **59**  **60**  **61**  **62**  **63**  **64**  **65**  **66**  **67**  **68**  **69**  **70**  **71**  **72**  **73** | **//3. Function**  **//reverse Function**  **int revNum(int num)**  **{**  **int rev = 0;**    **if (num < 10)**  **{**  **rev = num \* 10;**  **return rev;**  **}**  **while (num != 0)**  **{**  **int digit = num % 10;**  **rev = rev \* 10 + digit;**  **num /= 10;**  **}**    **return rev;**  **}//reverse ends here**      **//-----------------Insert-------------------------**  **void Hash :: Insert(int key)**  **{**  **//Hash fnc**  **int addr = (revNum((key / 10000) % 100) + ((key / 100) % 100) + revNum(key%100)) % 100;**  **int oaddr = addr;**    **if (h[addr] == EMPTY) //addr is available**  **{**  **h[addr] = key;**  **cout<<key<<" is inserted at the address: "<<addr<<endl;**  **}**  **else{**  **//Colliison**  **do**  **{** |

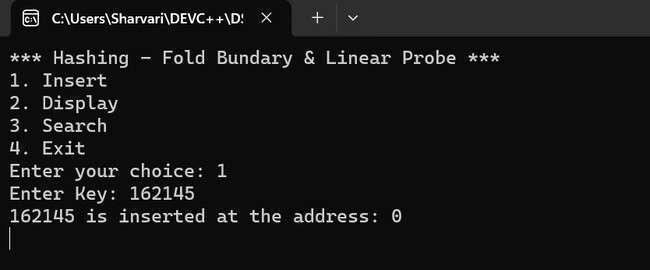
|  |  |
| --- | --- |
| **74**  **75**  **76**  **77**  **78**  **79**  **80**  **81**  **82**  **83**  **84**  **85**  **86**  **87**  **88**  **89**  **90**  **91**  **92**  **93**  **94**  **95**  **96**  **97**  **98**  **99**  **100**  **101**  **102**  **103**  **104**  **105**  **106**  **107**  **108**  **109**  **110**  **111**  **112**  **113**  **114** | **addr = (addr+1) % MAX;**    **if (addr == oaddr)**  **{**  **cout<<"Hash table is Full";**  **return;**  **}**  **}while(h[addr] != EMPTY);**    **h[addr] = key;**    **cout<<key<<" is inserted at the address: "<<addr<<endl;**  **}// end of if-else**  **}//endo of insert**      **//-----------------Display-------------------------**  **void Hash :: Display()**  **{**  **for (int i=0; i<MAX; i++)**  **{**  **if (h[i] == EMPTY)**  **{**  **cout<<"arr["<<i<<"]: "<<"EMPTY"<<endl;**  **continue;**  **}**  **cout<<"arr["<<i<<"]: "<<h[i]<<endl;**  **}**  **}//end of Display**      **//-----------------Search-------------------------**  **void Hash :: Search(int key)**  **{**  **int saddr = key%MAX;**  **int oaddr = saddr;**  **int i;**  **if (h[saddr] == key)**  **{**  **cout<<"key found at address: "<<saddr<<endl;**  **} else** |

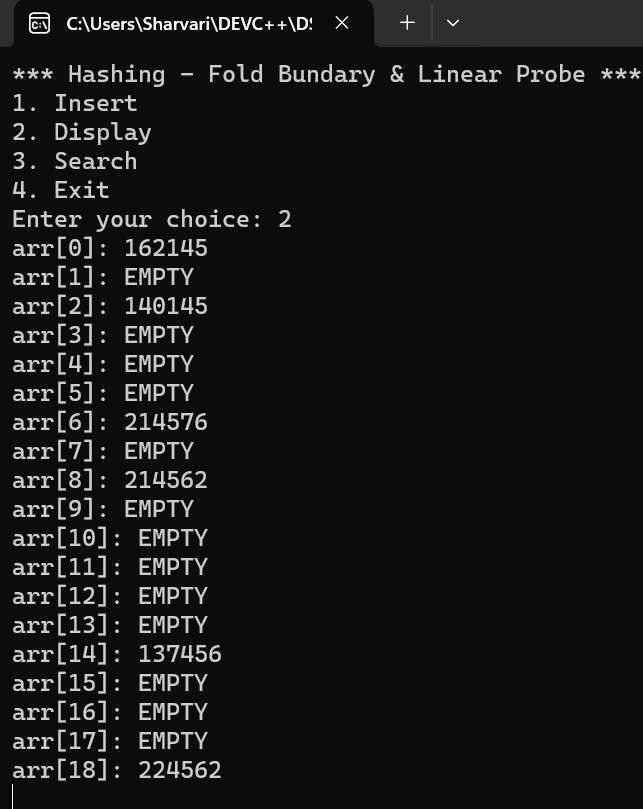
|  |  |  |  |
| --- | --- | --- | --- |
| **115**  **116**  **117**  **118**  **119**  **120**  **121**  **122**  **123**  **124**  **125**  **126**  **127**  **128**  **129**  **130**  **131**  **132**  **133**  **134**  **135**  **136**  **137**  **138**  **139**  **140**  **141**  **142**  **143**  **144**  **145**  **146**  **147**  **148**  **149**  **150**  **151**  **152**  **153**  **154**  **155** | **}**      **//4. Main Me**  **int main()**  **{**  **Hash h; int cho**    **while(t**  **{**                    **switch(** | **{**                                  **}**  **nu**    **ice, num; rue)**  **system( cout<< cout<< cout<< cout<< cout<<**    **cout<<**  **cin>>choice;**    **choice)**  **{** | **do**  **{**  **saddr = (saddr+1) % MAX;**  **if (saddr == oaddr)**  **{**  **break;**  **}**  **}while(h[saddr] != key);**    **if (h[saddr] == key)**  **{**  **cout<<"Key found at address: "<<saddr<<endl;**  **} else**  **{**  **cout<<"Key not found"<<endl;**  **}**    **"cls");**  **"\*\*\* Hashing - Fold Bundary & Linear Probe \*\*\*"<<endl;**  **"1. Insert"<<endl;**  **"2. Display"<<endl;**  **"3. Search"<<endl;** **"4. Exit"<<endl;**  **"Enter your choice: ";** |
| **156**  **157**  **158**  **159**  **160**  **161**  **162**  **163**  **164**  **165**  **166**  **167**  **168**  **169**  **170**  **171**  **172**  **173**  **174**  **175**  **176**  **177**  **178**  **179**  **180**  **181**  **182** | **case 1:**  **cout<<"Enter Key: ";**  **cin>>num;**  **h.Insert(num);**  **getch();**  **break;**  **case 2:**  **h.Display();**  **getch();**  **break;**  **case 3:**  **cout<<"Enter key to Search: ";**  **cin>>num;**  **h.Search(num);**  **getch();**  **break;**  **case 4:**  **exit(true);**    **default:**  **cout<<"Incorrect choice:(";**  **getch();**  **break;**  **} // end of switch**  **} // end of while**  **} // end of main** | | |

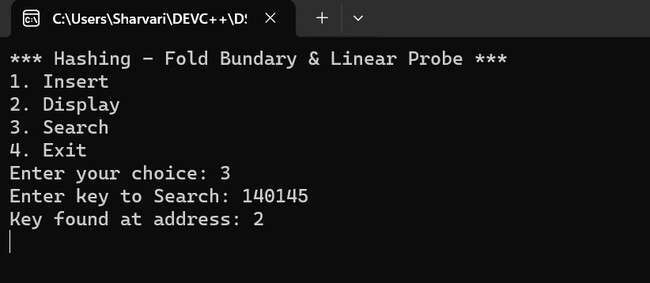
**Output:**











## Unit - 3 : Stacks

|  |  |
| --- | --- |
| **Program No:** | **12.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Array Based Stack** |
| **Objective :** | **Unit-3: Stacks** |

**Date:** 30.10.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\*Name: Sharvari Birajdar  Roll No: 1409  Unit 3: Stacks  Program: Array-Based Stack \*/    #include<iostream>  #include<conio.h>    #define SIZE **5** // Declaring a constant SIZE **using** **namespace** std;    /\*----------1. Node Template - NOT REQUIRED FOR ARRAY BASED STRUCTURE------  --\*/    /\*----------2. Array Based Template--------\*/ **class** **AStack**  {  **int** A[SIZE];  **int** tos;    **public:** AStack()  {  tos = -**1**; //Empty Stack  }    **void** Push(**int** x); **void** **Pop**(); **void** **Peek**(); **void** **Display**(); |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **int** **Full**();  **int** **Empty**();    };      /\*---------------3.Functions--------------\*/    /\*----------------FULL()---------------\*/ **int** AStack :: Full()  {  **if**(tos == SIZE-**1**)  {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of Full      /\*----------------Empty()---------------\*/  **int** AStack :: Empty()  {  **if**(tos == -**1**)  {  **return** **1**;  } **else** {  **return** **0**;  }    }//end of Empty      /\*----------------Push()---------------\*/  **void** AStack :: Push(**int** x)  {  **if**(Full()) |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | {  cout << "Stack Overflow"; **return**;  }    tos++; A[tos] = x;  cout << "Element inserted successfully!";    }//end of Push      /\*----------------Peek()---------------\*/  **void** AStack :: Peek()  {  **if**(Empty())  {  cout << "Stack Underflow!"; **return**;  }    cout << "Element at the TOS is: " << A[tos];    }//end of Peek      /\*----------------Display()---------------\*/  **void** AStack :: Display()  { **if**(Empty())  {  cout << "Stack underflow!"; **return**;  } **int** i;  cout<<"The Stack is: **\n**";  **for**(i=tos;i>=**0**;i--)  {  cout << A[i] <<endl;  } |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | }//end of Display    /\*----------------Pop()---------------\*/  **void** AStack :: Pop()  {  **if**(Empty())  {  cout << "Stack Underflow !!!"; **return**;  }    **int** temp = A[tos]; tos--;    cout << "Element popped is: " << temp;    }//end of pop      /\*---------------4.Menu--------------\*/    **int** main()  {  AStack s;  **int** ch,num;    **while**(**1**)  { system("cls");    cout << "\*\*\* Array-Based Stack \*\*\***\n\n**";    cout << "1. Push an element on the stack**\n**"; cout << "2. Pop the element at the TOS**\n**"; cout << "3. Peek at the element on the TOS**\n**"; cout << "4. Display the stack**\n**"; cout << "5. Exit**\n\n**";    cout<<"Enter your choice: "; |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191 | cin>>ch;    **switch**(ch)  { **case** **1**:  cout<<"Enter the element: "; cin>>num;  s.Push(num);    getch(); **break**;  **case** **2**:  s.Pop(); getch(); **break**;  **case** **3**:  s.Peek(); getch(); **break**;  **case** **4**:  s.Display(); getch(); **break**;  **case** **5**:  exit(**1**);  **default:** cout<<"Incorrect choice!!!"; getch();    }//end of switch    }//end of while  }//end of main |

**Output:**

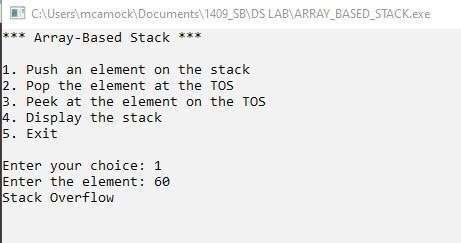
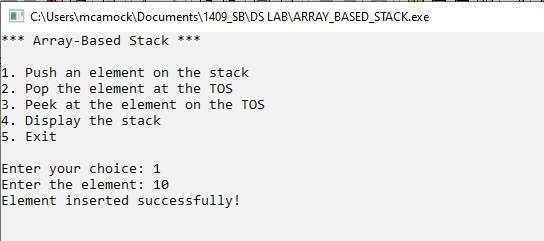
**(**

Push)

(

Pop

)

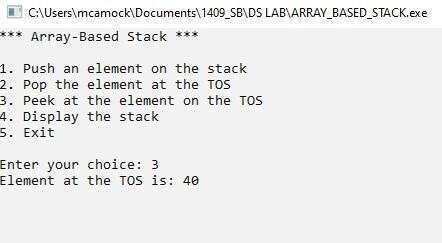
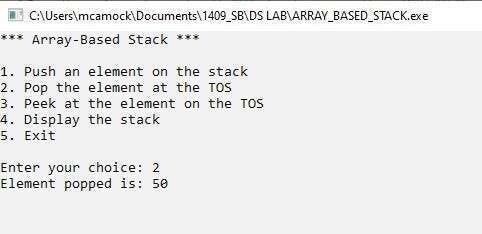


(

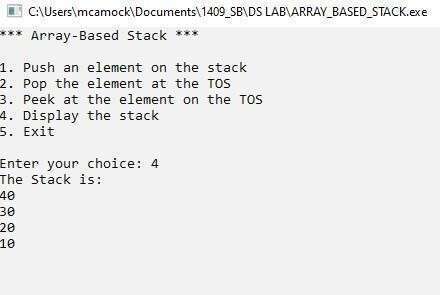
Peek

at Top of Stack

)



### (Display)



|  |  |
| --- | --- |
| **Program No:** | **13.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **List Based Stack** |
| **Objective :** | **Unit-3: Stacks** |

**Date:** 04.11.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* Name: Sharvari Birajdar  Roll no:1409  Unit 3: Stacks  Program: List-Based Stack \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;    /\*----------------1.Node Template------------------------\*/ **class** **SNode**  { **public:**  **int** data;  SNode \*next;  };    /\*-------------------2.List-Based Template--------------------\*/ **class** **LStack**  {  SNode \*tos;    **public:** LStack()  {  tos = NULL;  }    **void** Push(**int** x); **void** **Pop**(); **void** **Peek**(); |

|  |  |
| --- | --- |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | **void** **Display**(); **int** **Empty**();  };      /\*----------------------3.Functions----------------------\*/  /\*-----------------Push()---------------------\*/ **void** LStack :: Push(**int** x)  {  //Make a new node t using x SNode \*t = **new** SNode; t->data = x;  t->next = NULL;    //First Node to the stack **if**(tos == NULL)  {  tos=t;  }  **else** //Attach t to the start of the stack  {  t->next = tos;  tos = t;  }  }//end of Push      /\*----------------Empty()---------------\*/  **int** LStack :: Empty()  {  **if**(tos == NULL)  {  **return** **1**;  } **else** {  **return** **0**;  }    }//end of Empty |

|  |  |
| --- | --- |
| 74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114 | /\*------------------Pop()-------------------\*/  **void** LStack :: Pop()  {  //check it is empty **if**(Empty())  {  cout<<"Stack Underflow"; **return**;  }    //otherwise pop the element  SNode \*temp=tos; //set a temp pointer to tos tos = tos->next; //Shift tos to the next node - goes to NULL for single node cout<<"Element popped: "<<temp->data; //Display the popped element **delete** temp;    }//end of Pop      /\*------------------Peek()-------------------\*/  **void** LStack :: Peek()  {  //check it is empty **if**(Empty())  {  cout<<"Stack Underflow"; **return**;  }    cout<<"Element at the TOS is: "<<tos->data;    }//end of peek    /\*------------------Display()-------------------\*/ **void** LStack :: Display()  {  //check it is empty **if**(Empty())  {  cout<<"Stack Underflow"; **return**; |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155 | }    SNode \*temp=tos;    **while**(temp!=NULL)  {  cout << "|"<< temp->data << "| " << endl; cout<<"----"<< ;  temp=temp->next;  }    }//end of Display      /\*------------------------4.Main Menu----------------------\*/ **int** main()  {  LStack s; **int** num,choice;  **while**(**1**)  { system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* List-Based Stack\*\*\***\n\n**" | |   cout << ;     |  |  |  |  | | --- | --- | --- | --- | | "1. Push an element in the stack**\n**" | | | | | "2. Pop the element at the tos**\n**" | | | ; | | "3. Peek at the stack**\n**" | | ;  ; | | "4. Display the stack**\n**" | | | "5. Exit**\n\n**" | ; |   cout << ; cout << cout << cout << cout <<    cout << "Enter your choice: ";  cin >> choice;    **switch**(choice)  { **case** **1**:  cout<<"Enter the element: "; cin>>num;  s.Push(num); getch(); **break**; |

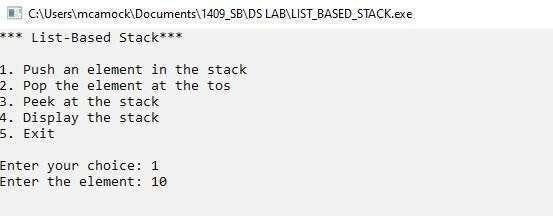
|  |  |
| --- | --- |
| 156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177 | **case** **2**:  s.Pop(); getch(); **break**; **case** **3**:  s.Peek(); getch(); **break**; **case** **4**:  s.Display(); getch(); **break**; **case** **5**:  exit(**1**); **default:** cout<<"Incorrect Choice!"; getch();  }//end of switch    }//end of while    }//end of main |

**Output:**

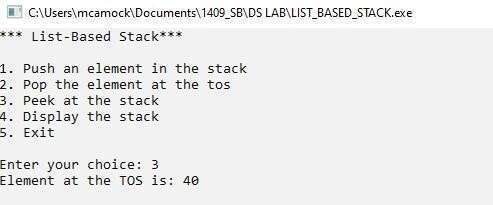
(

Push

)



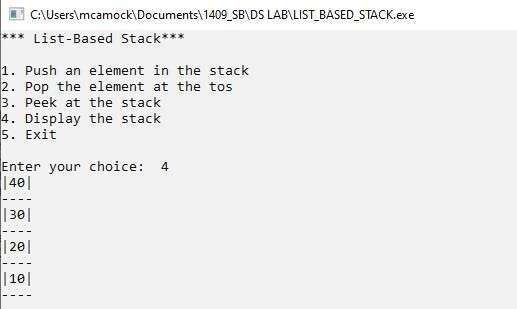
### (Peek)



(

Display

)



|  |  |
| --- | --- |
| **Program No:** | **14.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Evaluation of Postfix Expression using Stacks** |
| **Objective :** | **Unit-3: Stacks** |

**Date:** 06.11.2023

**Source Code:**

**(F\_Stack.cpp) :**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 3: Stacks  Program: Stack Template for Evaluation of Postfix Expression \*/    #include<iostream>  #include<conio.h>  #include<ctype.h>    **using** **namespace** std;  #define SIZE **20**    /\*Create a Stack\*/ **class** **Stack**  {  **float** A[SIZE];  **int** tos;    **public:** Stack()  { tos=-**1**;  }    **void** Push(**float** x); **float** **Pop**(); **int** **Full**();  **int** **Empty**();  };    /\*-----------------Full()---------------------\*/ |

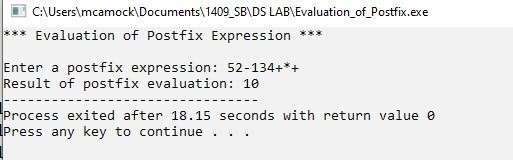
|  |  |
| --- | --- |
| 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | **int** Stack :: Full()  {  **if**(tos==SIZE-**1**)  {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of Full    /\*-----------------Empty()---------------------\*/ **int** Stack :: Empty()  {  **if**(tos==-**1**)  {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of Empty      /\*-----------------Push()---------------------\*/ **void** Stack :: Push(**float** x)  { **if**(Full())  {  cout<<"Stack Overflow";  **return**;  } tos++; A[tos]=x;  }//end of Push    /\*-----------------Pop()---------------------\*/ **float** Stack :: Pop() { |
| 73  74  75  76  77  78  79  80  81  82  83 | **if**(Empty())  {  cout<<"Stack Underflow";  **return** -**1** ;  }  **float** temp = A[tos]; tos--;  **return** temp;    }//end of Push |

**(Evaluation\_Of\_Postfix.cpp):**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 3: Stacks  Program: Evaluation of Postfix Expression \*/    #include"F\_Stack.cpp"    **int** main()  {  **char** postfix[**20**]; Stack s;  **float** result,n1,n2;  **int** i=**0**;    cout<<"\*\*\* Evaluation of Postfix Expression \*\*\***\n\n**";    cout<<"Enter a postfix expression: "; cin>>postfix;    **while**(postfix[i]!='\0')  {  **if**(postfix[i] == ' ')  { i++; **continue**;  } |
| 28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | **if**(isdigit(postfix[i]))  {  **float** x;  x = postfix[i] - **48.0**; s.Push(x); }  **else** //Operator  {  n2 = s.Pop(); n1 = s.Pop();    **switch**(postfix[i])  { **case** '+': result = n1 + n2; **break**; **case** '-': result = n1 - n2; **break**; **case** '\*': result = n1 \* n2; **break**; **case** '/': result = n1 / n2; **break**; }//end of switch    s.Push(result);    }//end of postfix[i] is an operator  i++;    }//end of while    cout << "Result of postfix evaluation: "<<s.Pop();    }//end of main |

**Output**

**:**



|  |  |
| --- | --- |
| **Program No:** | **15.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Balancing of Parenthesis using Stacks** |
| **Objective :** | **Unit-3: Stacks** |

**Date:** 09.11.2023

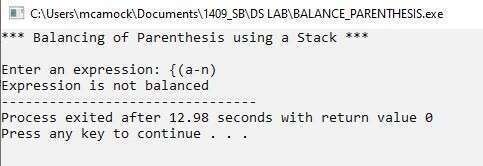
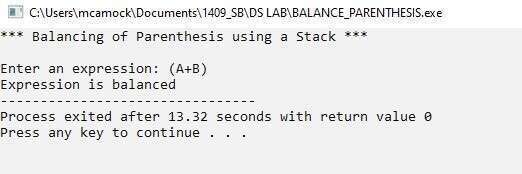
**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\* Name: Sharvari Birajdar  Roll no:1409  Unit 5: Stacks  Program: Balancing of Parenthesis\*/    #include<iostream>    **using** **namespace** std;  #define SIZE **20**    /\* Create a stack \*/ **class** **Stack**  {  **char** A[SIZE]; **int** tos;    **public:** Stack()  {  tos=-**1**;  }    **void** Push(**char** x); **char** **Pop**(); **int** **Full**(); **int** **Empty**();  };    /\*-----------------Push()---------------------\*/  **void** Stack :: Push(**char** x)  {  **if**(Full()) |

|  |  |
| --- | --- |
| 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73 | {  cout<<"Stack Overflow";  **return**;  }    tos++; A[tos] = x;  }//end of Push    /\*-----------------Pop()---------------------\*/  **char** Stack :: Pop()  {  **if**(Empty())  {  cout<<"Stack Undeflow";  **return** -**1**;  }    **char** temp = A[tos];  tos--;  **return** temp;    }//end of Pop    /\*-----------------Full()---------------------\*/ **int** Stack :: Full()  {  **if**(tos== SIZE-**1**)  { **return** **1**;  } **else** {  **return** **0**;  }  }//end of Full    /\*-----------------Empty()---------------------\*/ **int** Stack :: Empty()  {  **if**(tos==-**1**) |

|  |  |
| --- | --- |
| 74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114 | {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of Empty        **int** main()  {  **char** expr[**20**]; //manage exp  **char** temp; //to hold **int** i=**0**,flag=**0**;  Stack s;    cout << "\*\*\* Balancing of Parenthesis using a Stack \*\*\***\n\n**";    cout << "Enter an expression: "; cin>>expr;    **while**(expr[i] != '\0')  {  **if**(expr[i] == '[' || expr[i]=='(' || expr[i]=='{')  {  s.Push(expr[i]);  }//end of if opening bracket    **if**(expr[i]==']' || expr[i]==')' || expr[i]=='}')  {  **if**(s.Empty()) //when extra close bracket  {  flag=**1**; **break**;  }//end of if stack Empty    temp = s.Pop();    **if**((temp=='[' && expr[i]==']') || |
| 115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144 | (temp=='(' && expr[i]==')') ||  (temp=='{' && expr[i]=='}'))  { i++; **continue**;  }//end of matching brackets **else** { flag=**1**;  **break**;  }//end of brackets do not match    }//end of if closing bracket  i++;      }//end of while    **if**(s.Empty() && flag==**0**)  {  cout<<"Expression is balanced";  } **else**  {  cout<<"Expression is not balanced";  }      }//end of main |

**Output:**



## Unit - 4 : Queues

|  |  |
| --- | --- |
| **Program No:** | **16.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Ordinary Queue** |
| **Objective :** | **Unit-4: Queues** |

**Date:** 09.11.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 4: queues  Program: Ordinary Queue \*/    #include<iostream>  #include<conio.h>    #define SIZE **4** // Declaring a constant SIZE **using** **namespace** std;    /\*----------1. Node Template - NOT REQUIRED FOR ARRAY BASED STRUCTURE---  -----\*/    /\*----------2. Array Based Template--------\*/ **class** **AQueue**  {  **int** A[SIZE]; **int** front;  **int** rear;    **public:** AQueue()  { front = -**1**; rear = -**1**;  }  **void** Enqueue(**int** x); **void** **Dequeue**(); **void** **PeekFront**(); |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **void** **PeekRear**(); **void** **Display**(); **int** **Full**(); **int** **Empty**();  };    /\*---------------3.Functions--------------\*/    /\*----------Enqueue----------\*/  **void** AQueue :: Enqueue(**int** x)  {  **if**(Full())  {  cout << "Queue Overflow!"; **return**;  }    rear++; A[rear] = x;    //First Element  **if**(front==-**1**)  {  front++;  }    }//end of Enqueue    /\*---------------Full-------------------\*/  **int** AQueue :: Full()  {  **return**(rear == SIZE-**1** ? **1** : **0**);  }//end of Full    /\*--------------Empty-----------------\*/  **int** AQueue :: Empty()  {  **return**(front == -**1** ? **1** : **0**);  }//end of Empty |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | /\*-----------------Dequeue()----------------\*/  **void** AQueue :: Dequeue()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }      **int** temp=A[front]; //hold element in the temp variable    **if**(front==rear) //Single element in the queue  {  front=rear=-**1**; //Empty queue  } **else** {  front++;  }    cout<<"Element removed is: "<<temp;  }//end of Dequeue    /\*-------------Display-------------\*/  **void** AQueue :: Display()  {  **if**(Empty())  {  cout << "Queue Underflow!"; **return**;  } **int** i;    cout<<"Queue contains: **\n**";  **for**(i=front; i<=rear; i++)  {  cout << A[i] <<" ";  }  }//end of Display |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | /\*-------------Peek Front-------------\*/  **void** AQueue :: PeekFront()  {  **if**(Empty())  {  cout <<"Queue Underflow!"; **return**;  }  cout<<"Element at the front is: "<<A[front];    }//end of Peek Front      /\*-------------Peek Rear-------------\*/  **void** AQueue :: PeekRear()  {  **if**(Empty())  {  cout <<"Queue Underflow!"; **return**;  }  cout<<"Element at the rear is: "<<A[rear];    }//end of Peek Rear          /\*---------------4.Main Menu---------------\*/    **int** main()  {  AQueue q; **int** ch,num;  **while**(**1**)  {  system("cls"); |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | cout << "\*\*\* Ordinary Queue\*\*\***\n\n**";    cout << "1. Enqueue an element**\n**"; cout << "2. Dequeue**\n**"; cout << "3. Peek Front **\n**"; cout << "4. Peek Rear **\n**"; cout << "5. Display the queue **\n**"; cout << "6. Exit **\n\n**";    cout << "Enter your choice: "; cin>>ch;    **switch**(ch)  { **case** **1**:  cout << "Enter an element: "; cin>>num;  q.Enqueue(num);  getch(); **break**;  **case** **2**:  q.Dequeue(); getch(); **break**;  **case** **3**:  q.PeekFront();  getch(); **break**;  **case** **4**:  q.PeekRear();  getch(); **break**;  **case** **5**:  q.Display(); getch(); **break**; |

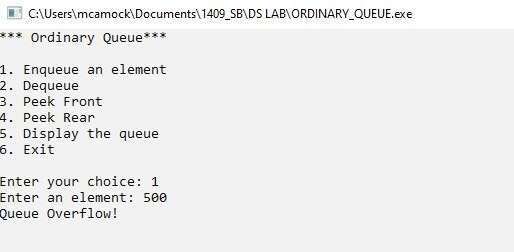
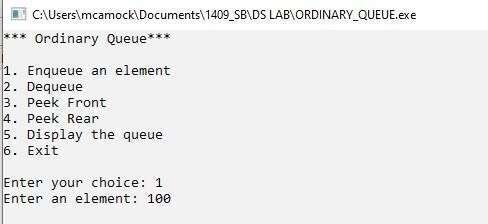
|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204 | **case** **6**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch();    }//end of switch  }//end of while  }//end of main |

**Output:**

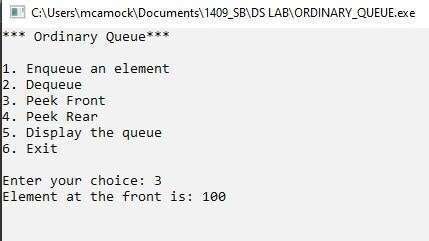
(

Enqueue

)



### (Peek Front)



(

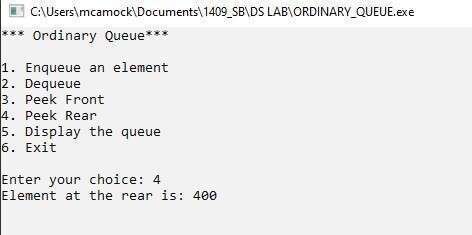
Peek Rear

)

(

Display

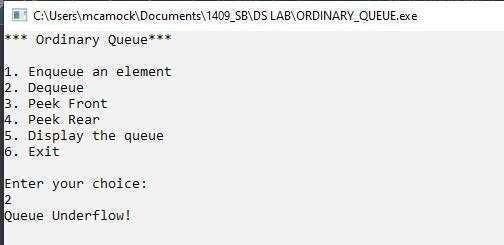
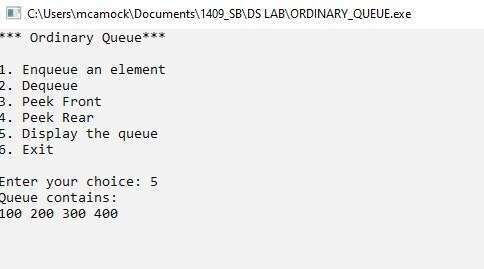
)



(

Dequeue

)



|  |  |
| --- | --- |
| **Program No:** | **17.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Circular Queue** |
| **Objective :** | **Unit-4: Queues** |

**Date:** 25.11.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 4: queues  Program: Circular Queue \*/    #include<iostream>  #include<conio.h>    #define SIZE 4 **using** **namespace** std;  /\*----------1. Node Template - NOT REQUIRED FOR ARRAY BASED STRUCTURE------  --\*/    /\*----------2. Array Based Template--------\*/    **class** **CQueue**  {  **int** A[SIZE]; **int** rear,front;  **int** count;    **public:** CQueue()  {  front=-**1**; rear=-**1**; count=**0**;  }    **void** Enqueue(**int** x); |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **void** **Dequeue**(); **void** **PeekFront**(); **void** **PeekRear**(); **void** **Display**(); **int** **Full**(); **int** **Empty**();  };      /\*---------------3.Functions--------------\*/    /\*---------------Full-------------------\*/  **int** CQueue :: Full()  {  **if**(count == SIZE)  {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of full    /\*---------------Empty-------------------\*/  **int** CQueue :: Empty()  {  **if**(count == **0**)  { **return** **1**;  } **else** {  **return** **0**;  }  }//end of empty      /\*---------------Enqueue(int x)-------------------\*/ **void** CQueue :: Enqueue(**int** x)  { |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **if**(Full())  {  cout<<"Queue Overflow!"; **return**;  }      **if**(rear == SIZE-**1**)//wrap rare index around the queue  {  rear = **0**;  } **else** {  rear++;  }    A[rear]=x;    **if**(front == -**1**) // Special case-first element in the queue  {  front++;  }  count++;    }//end of Enqueue      /\*-----------------Dequeue()----------------\*/  **void** CQueue :: Dequeue()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }    **int** temp=A[front];  **if**(front == rear) //Single Element Deletion  {  front=-**1**; rear=-**1**; |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | }  **else** **if**(front == SIZE-**1**) //Implement Circularity and update front  {  front=**0**;  } **else** {  front++;  }    cout << "Element removed: " << temp; count--;    }//end of Dequeue    /\*-----------------Display()----------------\*/  **void** CQueue :: Display()  {  **if**(Empty())  {  cout << "Queue Underflow!"; **return**;  }    **int** i=front;  **for**(**int** j=**1**; j<=count;j++)  {  cout << A[i] << "|";  **if**(i== SIZE-**1**)  { i=**0**;  } **else** {  i++;  }  }      }//end of Display |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | /\*-----------------PeekFront()----------------\*/ **void** CQueue :: PeekFront()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }    cout<<"Element at the front is: "<<A[front];  }//end of PeekFront    /\*-----------------PeekRare()----------------\*/  **void** CQueue :: PeekRear()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }    cout<<"Element at the front is: "<<A[rear];    }//end of PeekRare    /\*---------------4.Main Menu---------------\*/    **int** main()  {  **int** ch,num;  CQueue c;    **while**(**1**)  {  system("cls");  cout << "\*\*\* Circular Queue \*\*\***\n\n**"; cout << "1. Enqueue Operation**\n**"; cout << "2. Dequque Operation**\n**"; cout << "3. PeekFront Operation**\n**"; cout << "4. PeekRare Operation**\n**"; |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | |  |  | | --- | --- | | "5. Display Operation**\n**" | | | "6. Exit **\n\n**" | ; |   cout << ; cout <<    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  { **case** **1**:  cout << "Enter an element: "; cin>>num;  c.Enqueue(num);  getch(); **break**;  **case** **2**:  c.Dequeue(); getch(); **break**;  **case** **3**:  c.PeekFront();  getch(); **break**;  **case** **4**:  c.PeekRear();  getch(); **break**;  **case** **5**:  c.Display(); getch(); **break**;  **case** **6**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch(); |

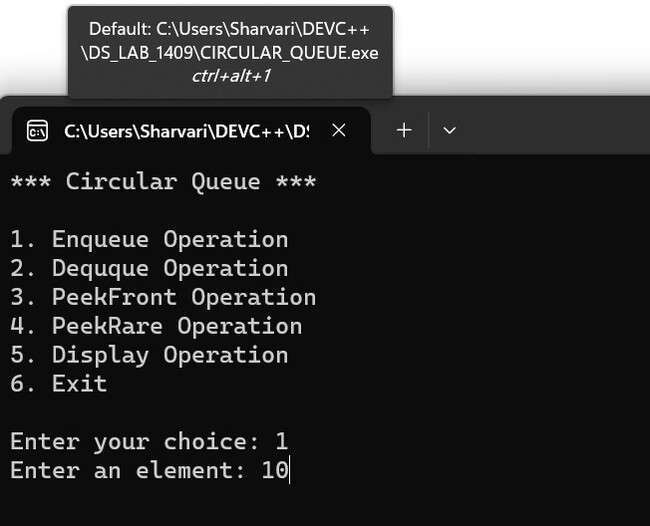
|  |  |
| --- | --- |
| 236  237  238  239 | }//end of switch    }//end of while  }//end of main |

**Output:**

(

Enqueue

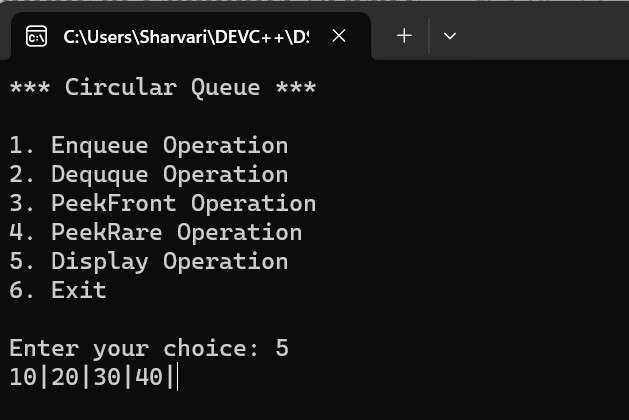
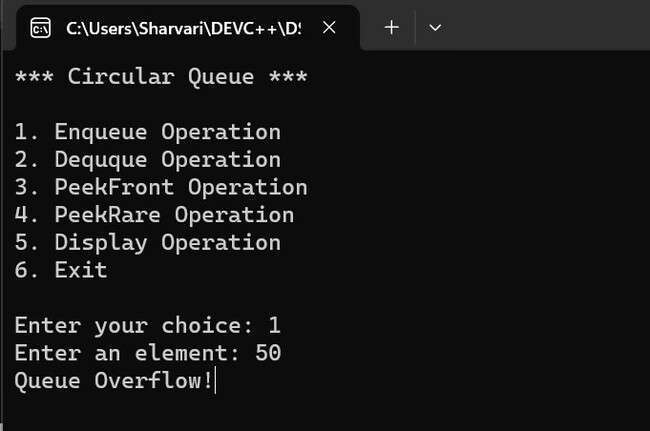
)



(

Display

)

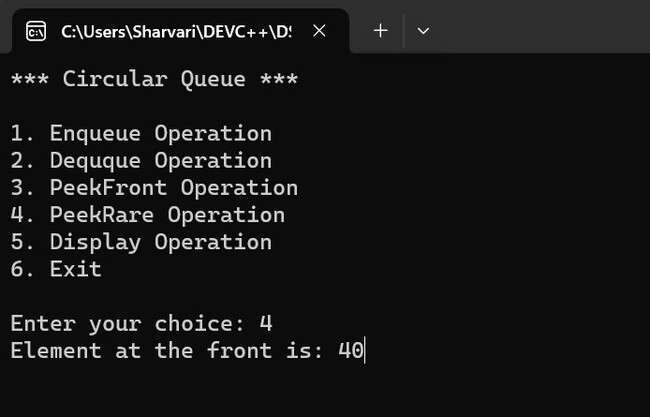
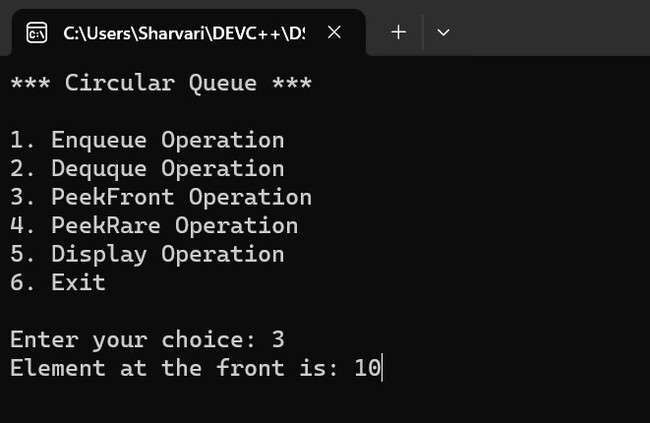


(Peek Front)

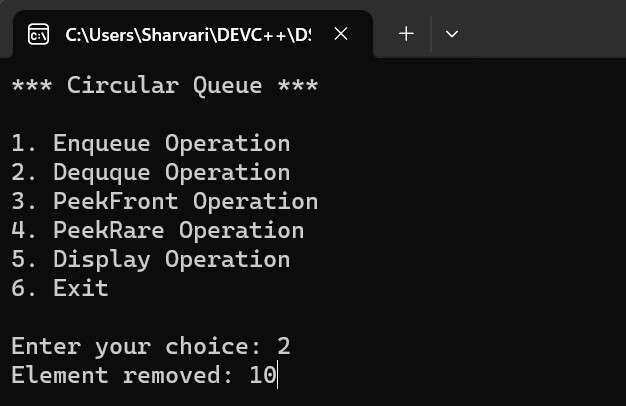
(

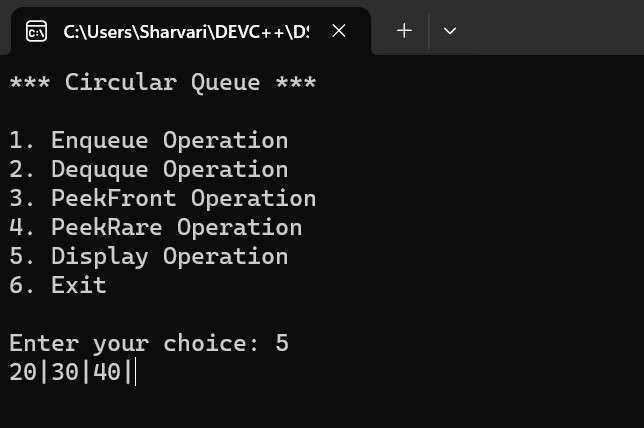
Peek Rare

)



(Dequeue)





|  |  |
| --- | --- |
| **Program No:** | **18.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Priority Queue** |
| **Objective :** | **Unit-4: Queues** |

**Date:** 04.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 4: queues  Program: Priority Queue \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;    /\*----------1. Node Template - NOT REQUIRED FOR ARRAY BASED STRUCTURE--------  \*/  **class** **PQNode**  { **public:**  **int** data;  **int** priority;  PQNode \*next;  };  //memory address initialization will be null    /\*----------2.List Template--------\*/  **class** **PQueue**  {  PQNode \*front; PQNode \*rear; **public:**    PQueue()  { |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | front=NULL;  rear=NULL;  }  **void** Enqueue(**int** x,**int** p); **void** **Dequeue**(); **void** **PeekFront**(); **void** **PeekRear**(); **void** **Display**(); **int** **Empty**();  };    /\*----------3.Functions-------\*/  /\*----------Enqueue-------------\*/ **void** PQueue :: Enqueue(**int** x,**int** p)  {  //1.Make a new Node  PQNode \*t = **new** PQNode; t->data=x;  t->priority=p;  t->next=NULL;    //2.1st Node in the queue **if**(front== NULL)  { front=t; rear=t; **return**;  }    //3.Traverse  PQNode \*temp= front;  PQNode \*prev=NULL;    **while**(temp!=NULL && temp->priority < t->priority)  {  prev=temp; temp=temp->next;  }    //4.Attach t  **if**(temp==front) //Front Node Insertion |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | {  t->next=front;  front=t;  }  **else** **if** (temp==NULL) //Rear Node Insertion  {  rear->next = t;  rear=t;  }  **else** //Any Other Node  {  prev->next=t;  t->next=temp;  }    }//end of Enqueue  /\*---------------Empty-------------------\*/  **int** PQueue :: Empty()  {  **if**(front == NULL)  {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of empty    /\*-----------------Display()----------------\*/  **void** PQueue :: Display()  {  **if**(Empty())  {  cout<<"Queue Underflow!!"; **return**;  }  PQNode \*temp=front;    **while**(temp!=NULL) |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | {  cout << "Data: " <<temp->data <<" Priority: "<<temp->priority<<endl; temp=temp->next;  }    cout<<"NULL";    }//end of Display    /\*-----------------PeekFront()----------------\*/  **void** PQueue :: PeekFront()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }    cout<<"Element at the front is: "<<endl;  cout << "Data: " <<front->data <<" Priority: "<<front->priority;    }//end of PeekFront    /\*-----------------PeekRear()----------------\*/  **void** PQueue :: PeekRear()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }    cout<<"Element at the rear is: "<<endl;  cout << "Data: " <<rear->data <<" Priority: "<<rear->priority;    }//end of PeekRear      /\*-----------------Dequeue()----------------\*/ |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | **void** PQueue :: Dequeue()  {  **if**(Empty())  {  cout<<"Queue Undeflow!"; **return**;  }  PQNode \*temp=front;  **if**(front==rear)  {  front = NULL;  rear = NULL;  } **else**  {  front=front->next;  }    cout <<"Element removed is : "<<temp->data<<" with priority " <<temp->priority;    **delete** temp;    }//end of Dequeue    /\*----------4.Menu--------\*/ **int** main()  {  **int** ch,num,pr;  PQueue p;    **while**(**1**)  {  system("cls");  cout << "\*\*\*Priority Queue\*\*\***\n\n**"; cout << "1. Enqueue Operation**\n**"; cout << "2. Dequeue Operation**\n**"; cout << "3. PeekFront Operation**\n**"; cout << "4. PeekRare Operation**\n**"; cout << "5. Display Operation**\n**"; cout << "6. Exit **\n\n**"; |

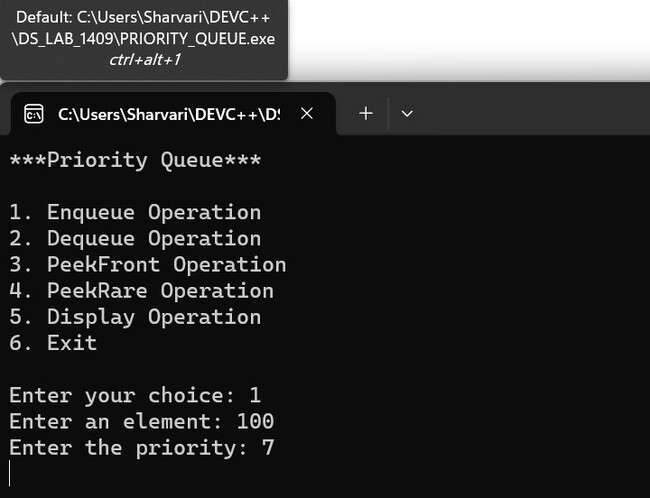
|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | cout << "Enter your choice: "; cin >> ch;  **switch**(ch)  { **case** **1**:  cout << "Enter an element: "; cin>>num;  cout << "Enter the priority: "; cin>>pr;  p.Enqueue(num,pr);  getch(); **break**;  **case** **2**:  p.Dequeue(); getch(); **break**;  **case** **3**:  p.PeekFront();  getch(); **break**;  **case** **4**:  p.PeekRear();  getch(); **break**;  **case** **5**:  p.Display(); getch(); **break**; **case** **6**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch();  } |
| 236  237  238 | }//end of while  }//end of main |

**Output:**

(

Enqueue

)

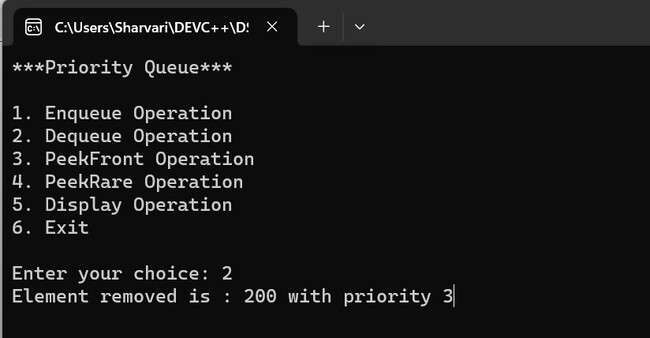
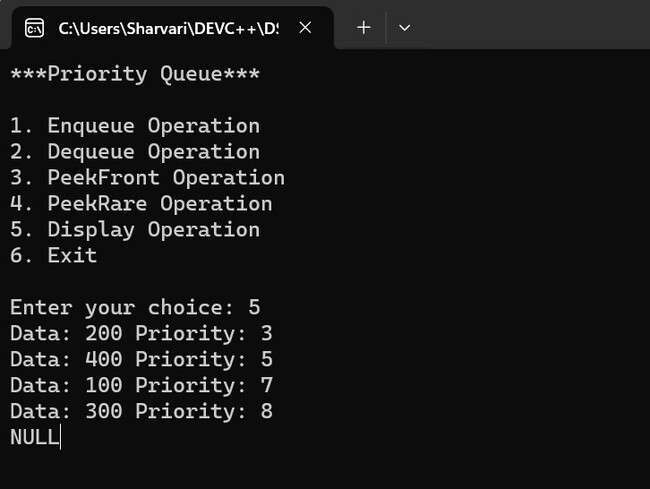


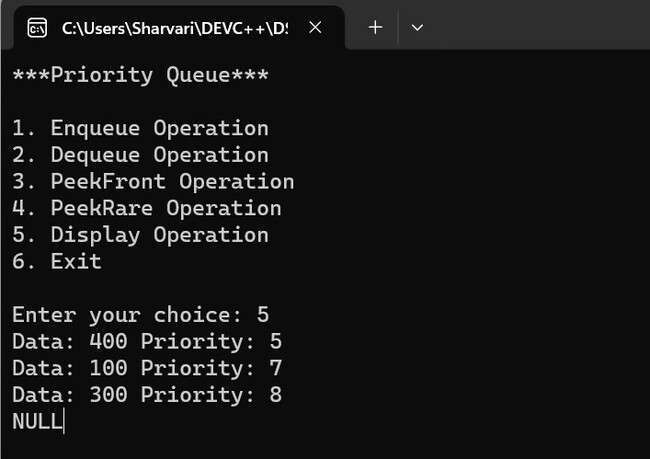
(Display)

(

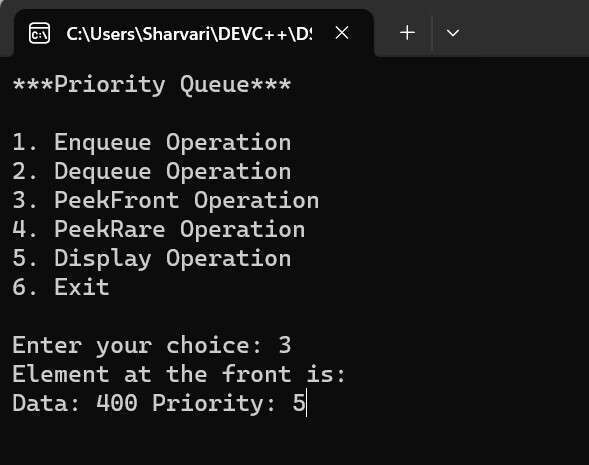
Dequeue

)

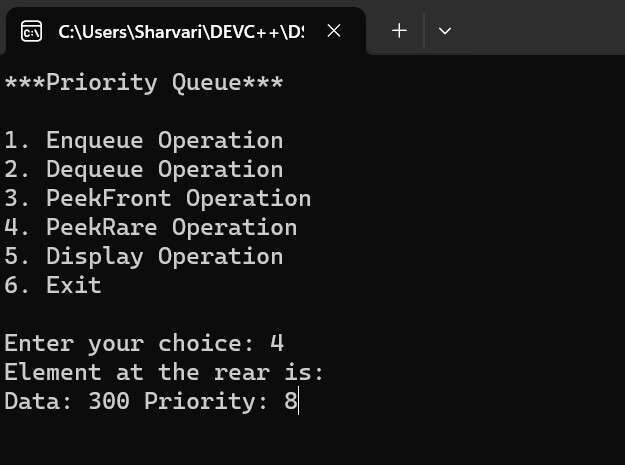




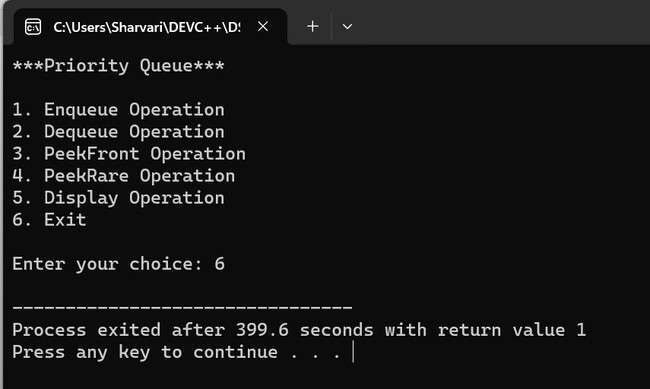
(Peek Front)



(Peek Rare)



(Exit)



|  |  |
| --- | --- |
| **Program No:** | **19.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Double-Ended Queue** |
| **Objective :** | **Unit-4: Queues** |

**Date:** 02.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 4: queues  Program: Double Ended Queues \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;  /\*----------1. Node Template -\*/  **class** **DQNode**  { **public:**  **int** data;  DQNode \*right;  DQNode \*left;  };    /\*----------2.List Template--------\*/  **class** **DQueue**  {  DQNode \*front;  DQNode \*rear;    **public**: DQueue()  {  front = rear = NULL;  } |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **void** EnqueueFront(**int** x); **void** **EnqueueRear**(**int** x); **void** **DequeueFront**(); **void** **DequeueRear**(); **void** **PeekFront**(); **void** **PeekRear**(); **void** **Display**(); **int** **Empty**();  };    /\*----------------3.Functions-------------\*/    /\*----------------EnqueueFront--------------\*/ **void** DQueue :: EnqueueFront(**int** x)  {  //Make a new node  DQNode \*t = **new** DQNode;  t->data = x; t->left = NULL;  t->right = NULL;    //Special case- 1st node in the queue **if**(front == NULL)  { front = t; rear = t; **return**;  }    //Attach t to the front of the queue t->right = front; front->left = t; front = t;    }//end of enqueue    /\*----------------EnqueueRear--------------\*/ **void** DQueue :: EnqueueRear(**int** x)  {  //Make a new node  DQNode \*t = **new** DQNode; |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | t->data = x; t->right = NULL;  t->left = NULL;    //1st Node in the queue  **if**(front == NULL)  { front = t; rear = t; **return**;  }    //Attach t to the rear of the queue rear->right = t; t->left = rear; rear = t;    }//end of EnqueueRear    /\*---------------Empty-------------------\*/  **int** DQueue :: Empty()  {  **if**(front == NULL)  {  **return** **1**;  } **else** {  **return** **0**;  }  }//end of empty    /\*--------------------Display()------------------\*/  **void** DQueue :: Display()  {  **if**(Empty())  {  cout << "Queue Underflow!"; **return**;  } |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | DQNode \*temp = front;    cout<< "The double ended queue is: " <<endl;    **while**(temp!=NULL)  {  cout << temp->data << " | "; temp = temp->right;  }    }//end of Display    /\*-------------------PeekFront()------------------\*/ **void** DQueue :: PeekFront()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }  cout << "Element at the front is: "<<front->data;      }//end of PeekFront    /\*-----------------PeekRear()----------------\*/  **void** DQueue :: PeekRear()  { **if**(Empty())  {  cout << "Queue Underflow!"; **return**;  }  cout << "Element at the front is: "<<rear->data;  }//end of PeekRear    /\*-----------------DequeueFront()----------------\*/  **void** DQueue :: DequeueFront()  {  **if**(Empty()) |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | {  cout<<"Queue Underflow!"; **return**;  }    DQNode \*temp = front; **if**(front == rear)  { front = rear = NULL;  } **else** {  front = front->right;  front->left=NULL;  }    cout << "Element removed is: " << temp->data << endl; **delete** temp;      }//end of DequeueFront    /\*-----------------DequeueRear()----------------\*/  **void** DQueue :: DequeueRear()  {  **if**(Empty())  {  cout<<"Queue Underflow!"; **return**;  }    DQNode \*temp = rear; **if**(front == rear)  { front = rear = NULL;  } **else** {  rear = rear->left;  rear->right=NULL;  } |

|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | cout << "Element removed is: " << temp->data << endl; **delete** temp;    }//end of DequeueRear    /\*----------4.Menu--------\*/ **int** main()  {  **int** ch,num;  DQueue d;    **while**(**1**)  {  system("cls");  cout <<" \*\*\*\* Double-Ended Queues \*\*\***\n\n**";  cout << "1. Enqueue Front**\n**"; cout << "2. Enqueue Rear**\n**"; cout << "3. Dequeue Front**\n**"; cout << "4. Dequeue Rear**\n**"; cout << "5. Peek Front**\n**"; cout << "6. Peek Rear**\n**"; cout << "7. Display**\n**"; cout << "8. Exit**\n**";    cout << "Enter your choice: "; cin >> ch;  **switch**(ch)  { **case** **1**:  cout << "Enter an element: "; cin>>num;  d.EnqueueFront(num); getch(); **break**;  **case** **2**:  cout << "Enter an element: "; cin>>num;  d.EnqueueRear(num); getch(); |

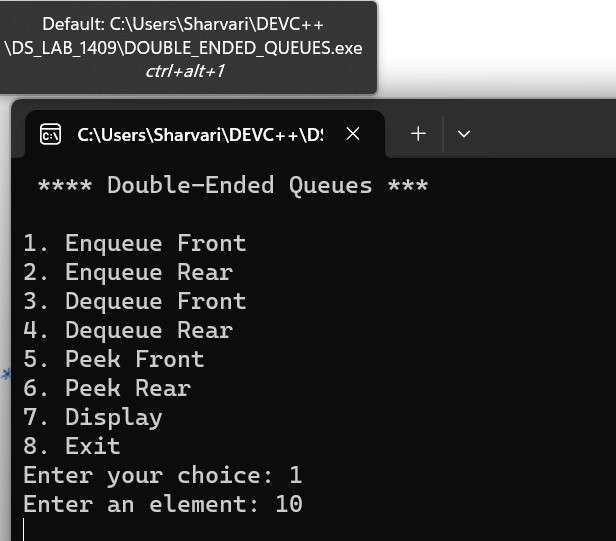
|  |  |
| --- | --- |
| 236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276 | **break**;  **case** **3**:  d.DequeueFront();  d.Display(); getch(); **break**;  **case** **4**:  d.DequeueRear();  d.Display(); getch(); **break**;  **case** **5**:  d.PeekFront();  getch(); **break**;  **case** **6**:  d.PeekRear();  getch(); **break**;  **case** **7**:  d.Display(); getch(); **break**;  **case** **8**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch();      }//end of switch    }//end of while  } |
| 277 |  |

**Output:**

(

Enqueue Front

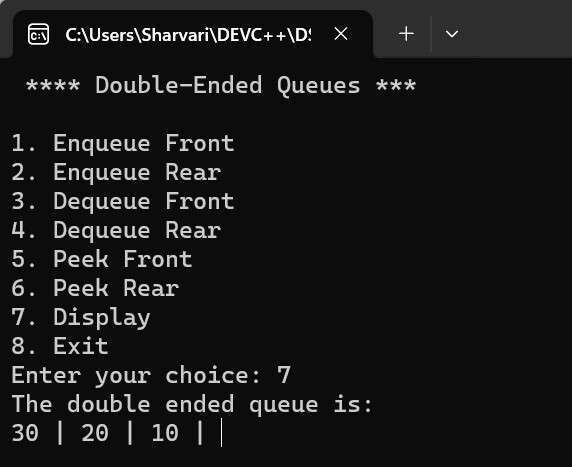
)

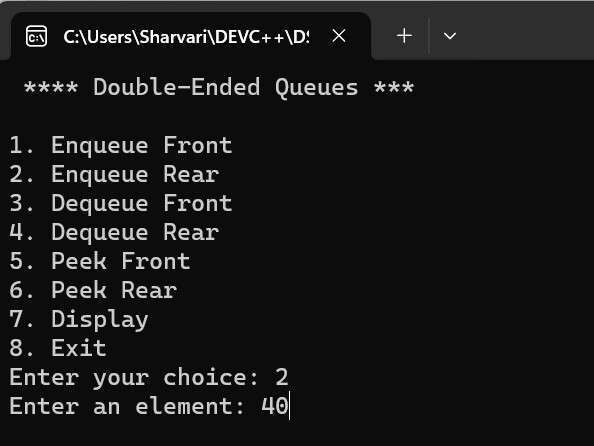


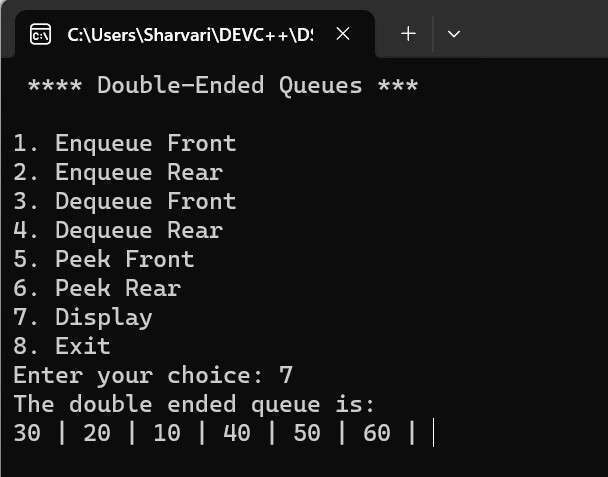
(

Enqueue Rare

)



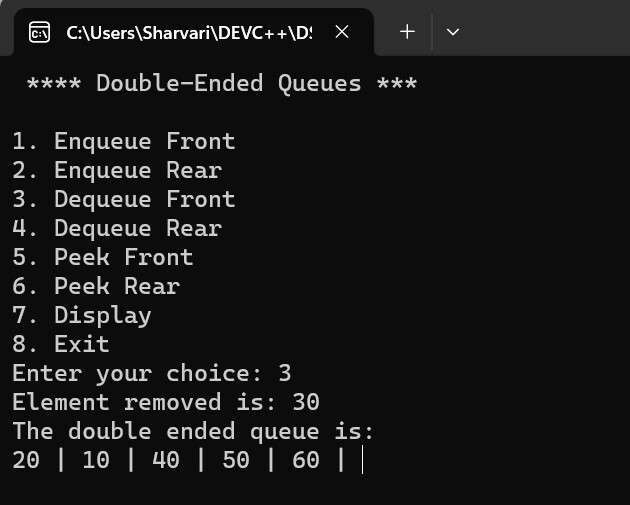




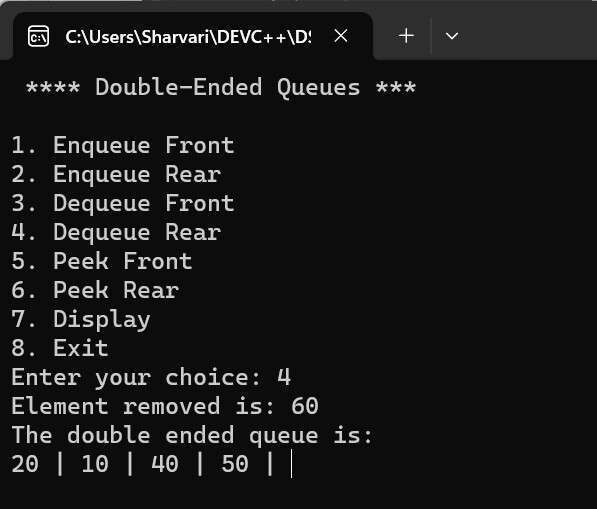
(

Dequeue Front

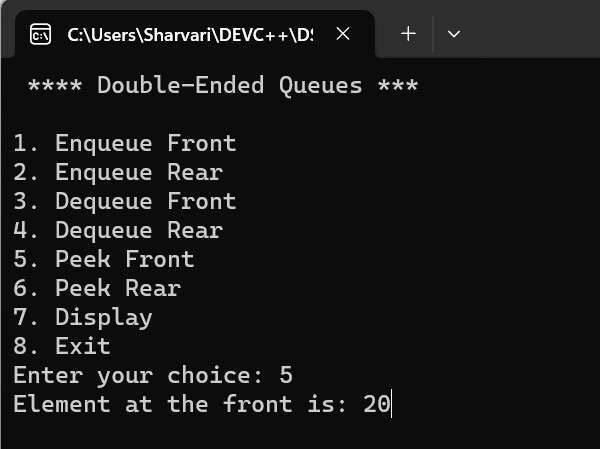
)



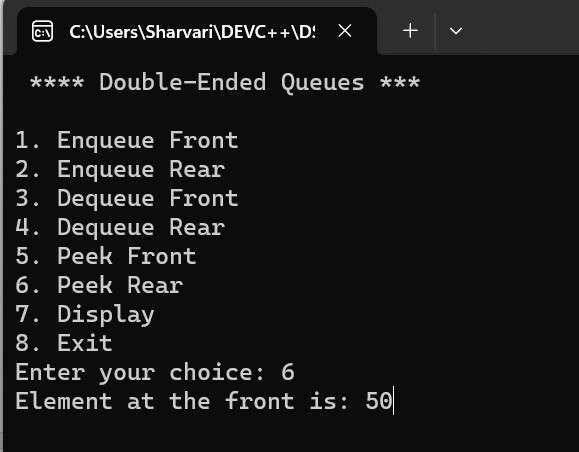
(Dequeue Rear)



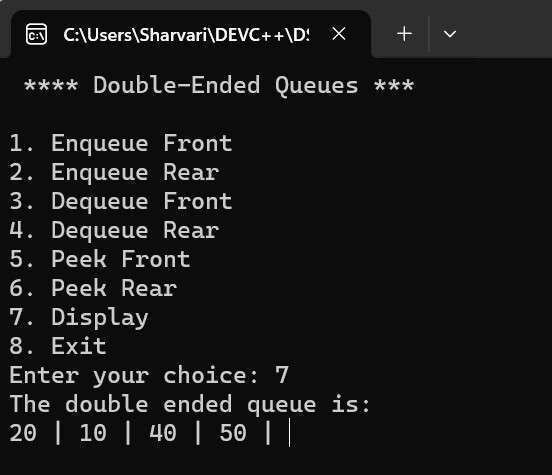
(Peek Front)



(Peek Rear)



(Display)



## Unit - 5 : Linked Lists

|  |  |
| --- | --- |
| **Program No:** | **20.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Singly Linked List** |
| **Objective :** | **Unit-5: Linked Lists** |

**Date:** 30.09.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll No: 1409  Unit 5:Linked Lists  Program: Singly Linked List \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;    /\*------------------1. Node Template------------------\*/ **class** **Node**  { **public:**  **int** data;  Node \*next;  };    /\*------------------2. List Template------------------\*/ **class** **List**  {  Node \*head;    **public**:  List()  {  head = NULL;  }  **void** Insert(**int** x); **void** **Del**(**int** x); |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **void** **Search**(**int** x); **void** **Length**(); **void** **Display**();  **void** **Reverse**();  };    /\*------------------3. Function------------------\*/ /\*-------------Insert Function---------------\*/ **void** List :: Insert(**int** x)  {  //Step-1: Make a new node t Node \*t = **new** Node;  t->data = x;  t->next = NULL;    //Step-2 Special Case: First node in the SLL **if**(head == NULL)  {  head = t;  }  **else** //Step-3: Traverse till the last node & attach t  {  Node \*temp = head; //Creating a temporary pointer    **while**(temp->next != NULL)  {  temp = temp->next;//assign the next address's node  }    temp->next = t;  }    }//end of insert    /\*-------------Display Function---------------\*/ **void** List :: Display()  {  Node \*temp = head;    **while**(temp != NULL)  { |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | cout <<temp->data << "->"; temp = temp->next;  }  cout << "NULL";  }//end of display    /\*-------------Length Function---------------\*/ **void** List :: Length()  {  Node \*temp = head;  **int** count=**0**;    **while**(temp != NULL)  {  count++;  temp = temp -> next;  }    cout << "Length of list is: " << count;  }//end of Length    /\*-------------Search Function---------------\*/ **void** List :: Search(**int** x)  {  Node \*temp = head;  **int** flag = **0**;  **int** pos=**0**;    **while**(temp != NULL)  {  pos++;  **if**(temp->data == x)  { flag = **1**; **break**;  }  temp = temp -> next;  }      **if** (flag==**1**) |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | {  cout << x << " is found at position " << pos ;  } **else**  {  cout << x << " is not found";  }  }//end of Search    /\*-------------Delete Function---------------\*/ **void** List :: Del(**int** x)  {    //1.Empty List - RETURN CONTROL  **if**(head==NULL)  {  cout << "Empty List!";  **return**;  }    //2. Search for x  Node \*temp = head;  Node \*prev = NULL;//first prev moves ,then temp moves **int** flag=**0**;      **while**(temp != NULL)  {  **if**(temp->data == x)  {  flag=**1**; **break**;  }  prev = temp;  temp = temp->next;  }    //3.Unsuccessful Search - RETURN CONTROL **if**(flag == **0**)  {  cout << x << " is not found"; |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | **return**;  }    //4.Successful Search  //a.Single node deletion  **if**(temp == head && temp->next==NULL)  {  head=NULL;  }  **else** **if** (temp==head)//b.head node deletion  {  head = temp->next;  }  **else** **if**(temp->next == NULL)//c.tail/last node deletion  {  prev->next = NULL;  }  **else**//d.delete any other node in the middle  {  prev->next = temp->next;  }  //5.Delete memory occupied by temp **delete** temp;    //Displaying the updated list after the node deletion  cout<<"The updated list is: "<<endl;  Display();    }//end of Del    /\*-------------Reverse Function---------------\*/ **void** List :: Reverse()  {  **if**(head == NULL)  {  cout << "Empty SLL";  **return**;  }    Node \*temp = head; |

|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | Node \*revHead = NULL;    //Head Node Insertion **while**(temp)  {  //create a new node with the contents of original SLL node  Node \*t = **new** Node; t->data = temp->data;  t->next = revHead; // for the first time node revHead is already NULL    //update revHead  revHead = t;    //Advance to th enect node in the original SLL temp = temp->next;  }    head = revHead; //set head to the reversed list    }//end of reverse    /\*------------------4. Main Menu------------------\*/ **int** main()  {  List s; **int** ch,num;    **while**(**1**)  { system("cls");    cout << "\*\*\* Singly Linked List \*\*\***\n\n**";    cout << "1. Insert a node" << endl; cout << "2. Delete a node" << endl; cout << "3. Search for a node" << endl; cout << "4. Length of the list" << endl; cout << "5. Display the list" << endl; cout << "6. Reverse the list" << endl; cout << "7. Exit" << endl << endl; |

|  |  |
| --- | --- |
| 236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276 | cout << "Enter your choice: "; cin >>ch;    **switch**(ch)  { **case** **1**:  cout << "Enter a value: "; cin >> num;    s.Insert(num);  getch(); **break**;  **case** **2**:  cout << "Enter the element to be deleted: "; cin >> num; s.Del(num); getch(); **break**;  **case** **3**:  cout << "Enter the element to be searched: "; cin >> num;  s.Search(num);  getch(); **break**;  **case** **4**:  s.Length(); getch(); **break**;  **case** **5**:  s.Display(); getch(); **break**;  **case** **6**:  s.Reverse(); cout<<"The reversed list is: "; s.Display(); |

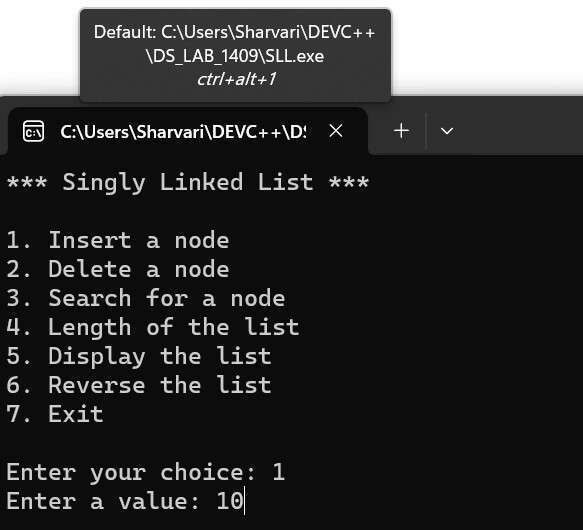
|  |  |
| --- | --- |
| 277  278  279  280  281  282  283  284  285  286  287  288  289 | getch(); **break**;  **case** **7**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch();    }//end of switch  }//end of while  }//end of main |

**Output:**

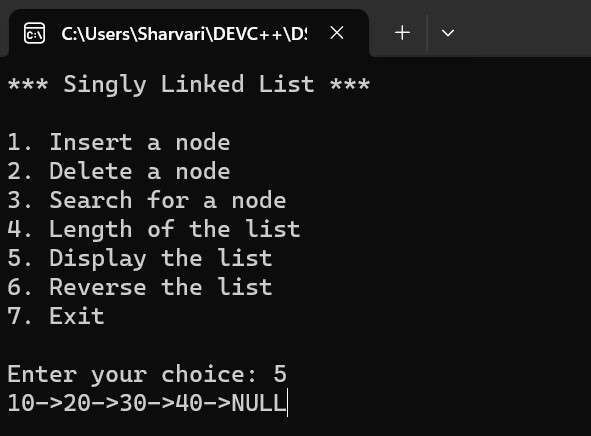
(

Insertion

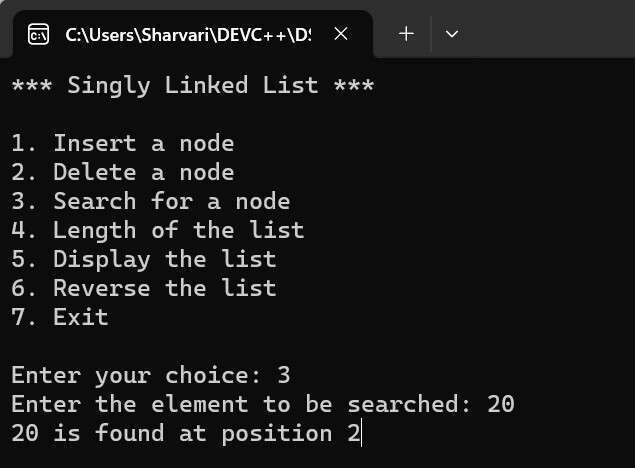
)



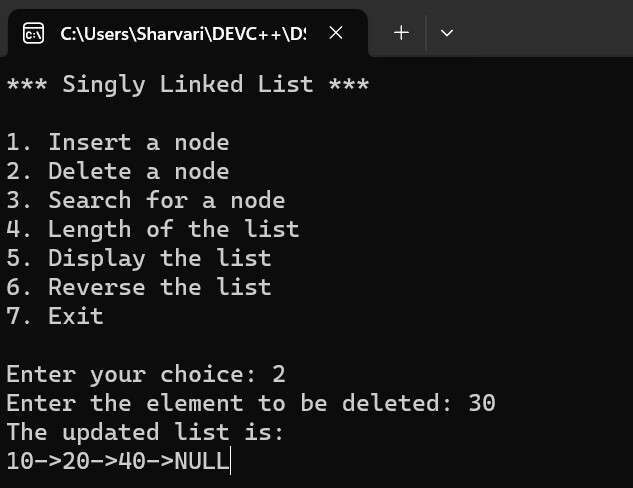
(Display)



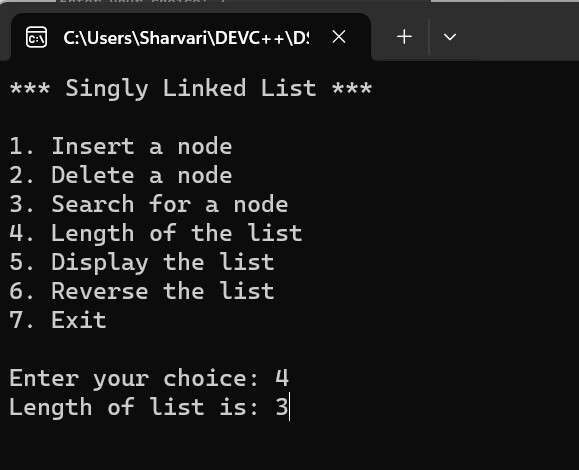
(Search)



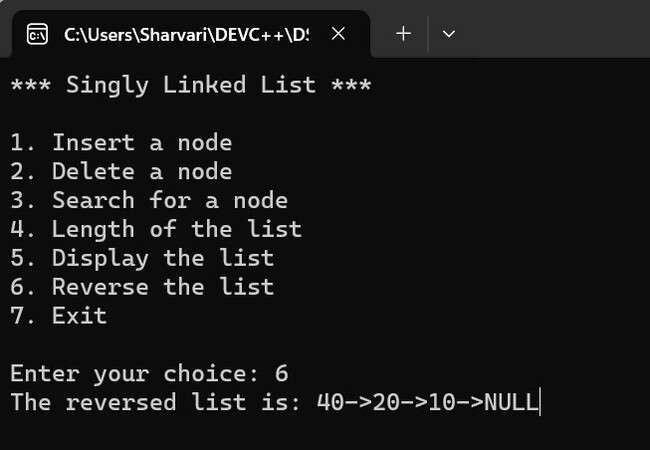
(Delete)



(Length)



(Reverse)



|  |  |
| --- | --- |
| **Program No:** | **21.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Circular Linked List** |
| **Objective :** | **Unit-5: Linked Lists** |

**Date:** 09.10.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\*Name: Sharvari Birajdar  Roll No: 1409  Unit 5:Linked Lists  Program: Circular Linked List \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;    /\*------------------1. Node Template------------------\*/ **class** **CNode**  { **public:**  **int** data;  CNode \*next;  };    /\*------------------2. List Template------------------\*/ **class** **CList**  {  CNode \*first;  CNode \*last;    **public:** CList()  {  first = last = NULL;  }  **void** Insert(**int** x); |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **void** **Del**(**int** x); **void** **Search**(**int** x); **void** **Display**();  **void** **Length**();    };    /\*------------------3. Function------------------\*/ //----------------Insert Function----------------- **void** CList :: Insert(**int** x)  {  //Make a new node t CNode \*t = **new** CNode; t->data = x;  t->next = NULL;    //First node in the CLL  **if**(first == NULL)  { first = t; last = t;  last->next = first;  }  **else** //Attach t to the last node  {  last->next = t; //Link last node to new node t last = t; //Shift last to t  last->next = first; //to maintain circularity  }  }//end of insert    //----------------Display Function----------------- **void** CList :: Display()  {  **if**(first == NULL)  {  cout << "Empty CLL!";  **return**;  }  CNode \*temp = first; |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **do**  {  cout << temp->data << "->"; temp = temp->next;  }  **while**(temp != first);  cout<<"Back to first node";  }//end of Display    //----------------Length Function----------------- **void** CList :: Length()  {    **if**(first == NULL)  {  cout << "Empty CLL!";  **return**;  }    CNode \*temp = first;  **int** count=**0**;  **do** {  count++;  temp=temp->next;    }  **while**(temp != first);    cout << "Number of nodes: " << count;    }//end of length      //----------------Search Function----------------- **void** CList :: Search(**int** x)  {  **if**(first == NULL)  {  cout<<"Empty CLL"; |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | **return**;  }    CNode \*temp=first;  **int** flag=**0**;  **int** pos=**0**;  **do** { pos++;  **if**(temp->data ==x)  { flag=**1**; **break**;  }  temp=temp->next;  }  **while**(temp != first);    **if** (flag==**1**)  {  cout << x << " is found at position " << pos ;  } **else**  {  cout << x << " is not found";  }  }//end of Search    //----------------Delete Function----------------- **void** CList :: Del(**int** x)  {  //1.Empty List - RETURN CONTROL **if**(first==NULL)  {  cout<<"Empty CLL"; **return**;  }    //2.Search for x  CNode \*temp=first; |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | CNode \*prev=NULL;  **int** flag=**0**;  **do**  {  **if**(temp->data == x)  { flag=**1**; **break**;  }    prev=temp; temp=temp->next;  }  **while**(temp!=first);    //3.Unsuccesful Search - RETURN CONTROL **if**(flag == **0**)  {  cout<<"Unsuccessful Search"; **return**;  }    //4.Successful Search //4a.Single Node Deletion  **if**(temp==first && temp==last)  {  first=NULL;  last=NULL;  }  //b.first node deletion (Update Cicularity) **else** **if** (temp==first)  {  first=first->next; last->next=first;  }  //c.last node deletion (Update Cicularity) **else** **if**(temp==last)  {  last=prev; last->next=first; |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | }  //d.delete any other node in the middle **else**  {  prev->next = temp->next;  }  //5.Delete memory occupied by temp **delete** temp;    //Displaying the updated list after the node deletion  cout<<"The updated list is: "<<endl;  Display();      }//end of Del  /\*------------------4. Main Menu------------------\*/ **int** main()  {  CList c;  **int** ch,num;    **while**(true)  { system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* Cicular Linked List Menu \*\*\***\n\n**" | |   cout << ;    cout << "1.Insert a node in the CLL**\n**"; cout << "2.Delete a node from the CLL**\n**"; cout << "3.Search a node in the CLL**\n**"; cout << "4.Length of the CLL**\n**"; cout << "5.Display the CLL**\n**";  cout << "6.Exit**\n\n**";    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  { **case** **1**:  cout << "Enter a value: "; cin >> num; |
| 236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270 | c.Insert(num);  getch(); **break**; **case** **2**:  cout << "Enter the element to be deleted: "; cin >> num; c.Del(num); getch(); **break**; **case** **3**:  cout << "Enter the element to be searched: "; cin >> num;  c.Search(num);  getch(); **break**; **case** **4**:  c.Length(); getch(); **break**; **case** **5**:  c.Display(); getch(); **break**; **case** **6**:  exit(**1**); **default:** cout << "Incorrect choice"; getch();    }//end of switch  }//end of while  }//end of main |

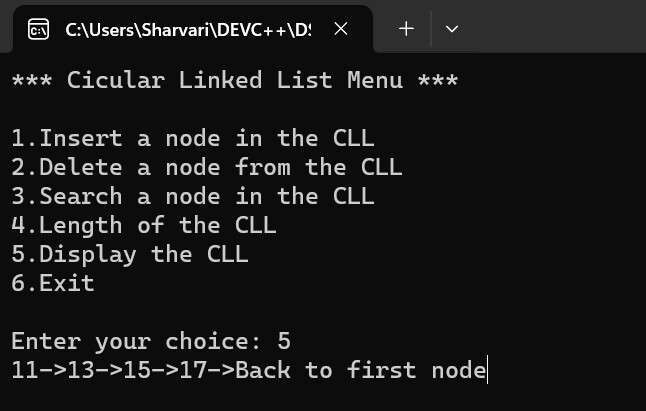
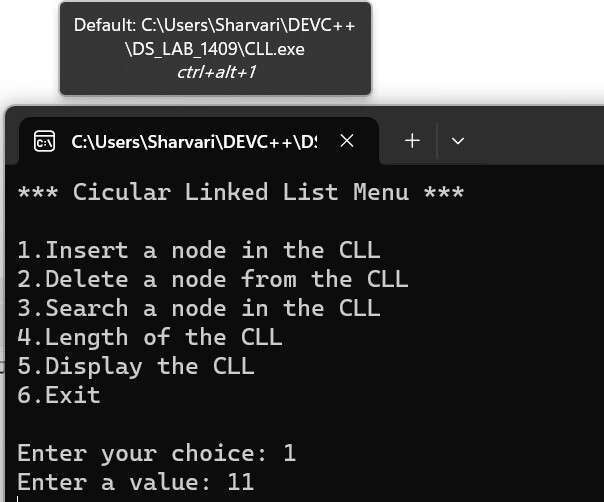
**Output:**

(Insertion)

(

Display

)

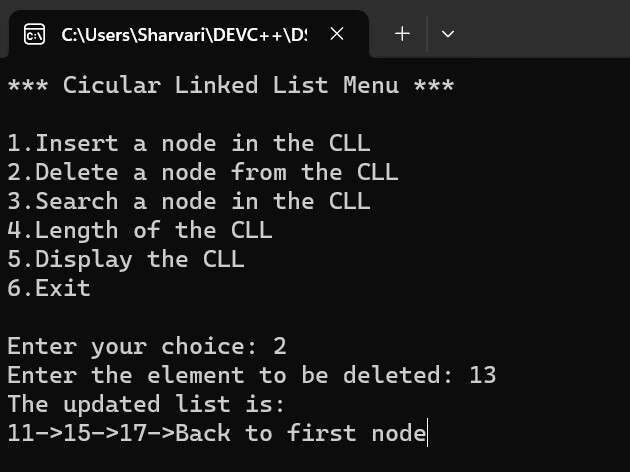
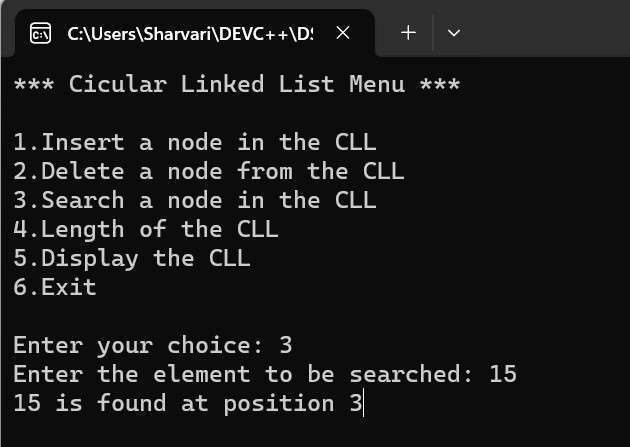


(Search)

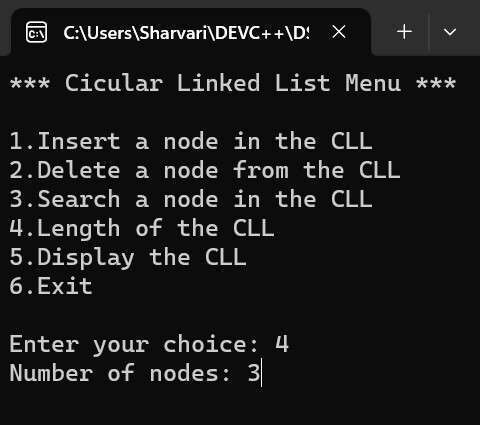
(

Delete

)



(Length)



|  |  |
| --- | --- |
| **Program No:** | **22.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Doubly Linked List** |
| **Objective :** | **Unit-5: Linked Lists** |

**Date:** 16.10.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\*Name: Sharvari Birajdar  Roll No: 1409  Unit 5:Linked Lists  Program: Doubly Linked List \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;    /\*------------------1. Node Template------------------\*/ **class** **DNode**  { **public:**  DNode \*left; **int** data;  DNode \*right;  };    /\*------------------2. List Template------------------\*/ **class** **DList**  {  DNode \*head;  DNode \*tail;    **public:** DList()  {  head = tail = NULL;  } |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **void** Insert(**int** x); **void** **Del**(**int** x); **void** **Search**(**int** x); **void** **Length**();  **void** **Display**();    };    /\*------------------3. Function------------------\*/ /\*-------------Insert Function---------------\*/ **void** DList::Insert(**int** x)  {  //Step-1: Make a new node t DNode \*t = **new** DNode; t->data = x; t->left = NULL;  t->right = NULL;    //Step-2 Special Case: First node in the DLL **if**(head == NULL)  {  head=t; tail=t;  }  **else** //when two nodes are there  {  tail->right=t; //attach t to the right of the tail t->left=tail; //attach tail to th left of t tail=t; //Shiting of tail  }    }//end of insert    /\*-------------Display Function---------------\*/ **void** DList :: Display()  {  DNode \*temp = head;     |  |  | | --- | --- | | "DLL in forward direction: " | | | "NULL<->" | ; |   cout << ; cout << |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **while**(temp != NULL)  {  // temp = temp->left; ....this will be NULL cout << temp->data << "<->";  temp = temp->right;  }  cout << "NULL**\n\n**";    //setting tail to display reverse direction temp = tail;   |  |  | | --- | --- | | "DLL in reverse direction: " | | | "NULL<->" | ; |   cout << ; cout <<  **while**(temp != NULL)  {  cout << temp->data << "<->";  temp = temp->left;  // temp = temp->right; ....this will be NULL  }    cout << "NULL";  }//end of display    /\*-------------Length Function---------------\*/ **void** DList :: Length()  {  DNode \*temp=head;  **int** count=**0**;    **while**(temp != NULL)  {  count++;  temp= temp->right;  }    cout << "Length of list is: " << count;  }//end of Length      /\*-------------Search Function---------------\*/ **void** DList :: Search(**int** x)  { |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | //1.Empty List - RETURN CONTROL  **if**(head==NULL && tail==NULL)  {  cout<<"Empty List!";  **return**;  }    //2.Search for x  DNode \*temp = head; **int** flag=**0**;  **int** pos=**0**;    **while**(temp != NULL)  { pos++;  **if**(temp->data == x)  { flag=**1**; **break**;  }  temp = temp->right;  }    **if**(flag == **1**)  {  cout << x << " is found at position in forward direction " <<pos;  } **else**  {  cout << x << " is not found";  }  }//end of Search    /\*-------------Del Function---------------\*/ **void** DList :: Del(**int** x)  {  //1.Empty List - RETURN CONTROL  **if**(head==NULL && tail==NULL)  {  cout<<"Empty List!"; **return**; |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | }    //2.Search for x  DNode \*temp = head; DNode \*prev = NULL;  **int** flag=**0**;    **while**(temp != NULL)  {  **if**(temp->data == x)  { flag=**1**; **break**;  }  prev = temp;  temp = temp->right;  }    //3.Unsuccessful Search - RETURN CONTROL **if**(flag==**0**)  {  cout << x <<" is not found"; **return**;  }    //4.Successful Search //a.Single node deletion  **if**(temp == head && temp == tail)  {  head=NULL;  tail=NULL;  }    //b.head node deletion  **else** **if**(temp == head)  {  head = head->right;  head->left= NULL;  }    //c.tail/last node deletion |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | **else** **if**(temp == tail)  {  tail = prev;  tail->right = NULL;  }    //d.delete any other node in the middle **else**  {  prev->right = temp->right; (temp->right)->left = prev;  }    //5.Delete memory occupied by temp **delete** temp;    //Displaying the updated list after the node deletion  cout<<"The updated list is: "<<endl;  Display();    }//end of Del  /\*------------------4. Main Menu------------------\*/    **int** main()  {  DList d;  **int** ch,num;    **while**(true)  { system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* Doubly Linked List \*\*\***\n\n**" | |   cout << ;    cout << "1. Insert a node" << endl; cout << "2. Delete a node" << endl; cout << "3. Search for a node" << endl; cout << "4. Length of the list" << endl; cout << "5. Display the list" << endl; cout << "6. Exit" << endl << endl;    cout << "Enter your choice: "; |

|  |  |
| --- | --- |
| 236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276 | cin >>ch;    **switch**(ch)  { **case** **1**:  cout << "Enter a value: "; cin >> num;    d.Insert(num);  getch(); **break**;  **case** **2**:  cout << "Enter the element to be deleted: "; cin >> num; d.Del(num); getch(); **break**;  **case** **3**:  cout << "Enter the element to be searched: "; cin >> num;  d.Search(num);  getch(); **break**;  **case** **4**:  d.Length(); getch(); **break**;  **case** **5**:  d.Display(); getch(); **break**;  **case** **6**:  exit(**1**);  **default:**  cout << "Incorrect Choice!"; |

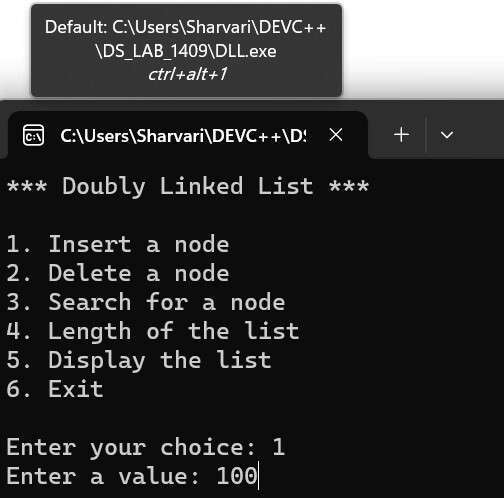
|  |  |
| --- | --- |
| 277  278  279  280  281  282 | getch();    }//end of switch  }//end of while    }//end of main |

**Output:**

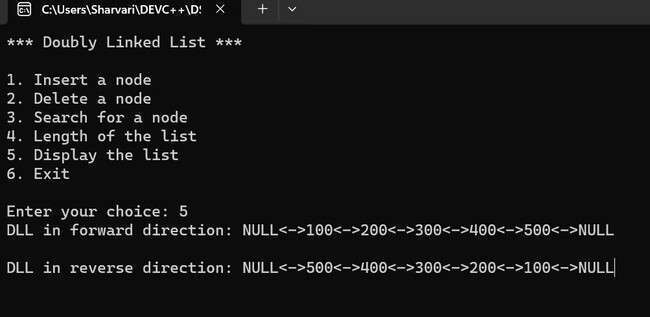
(

Insertion

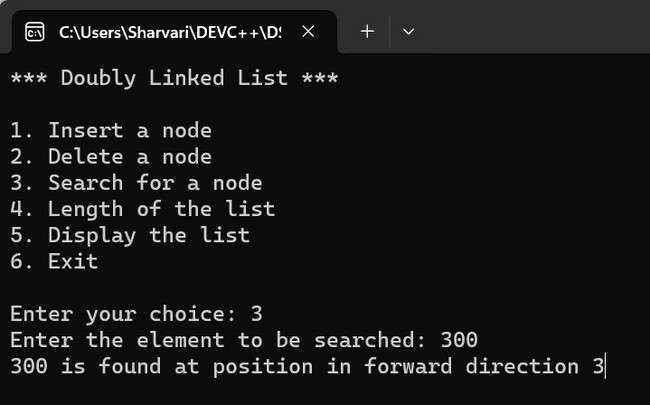
)



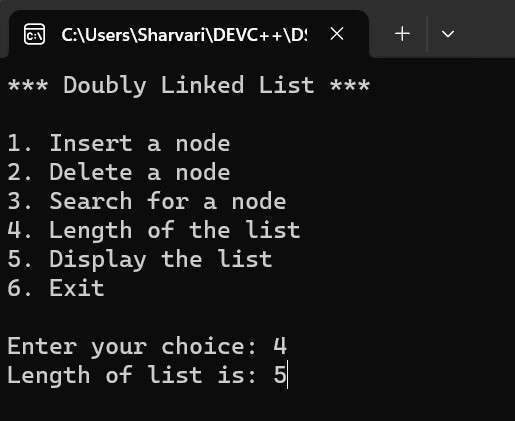
(Display)



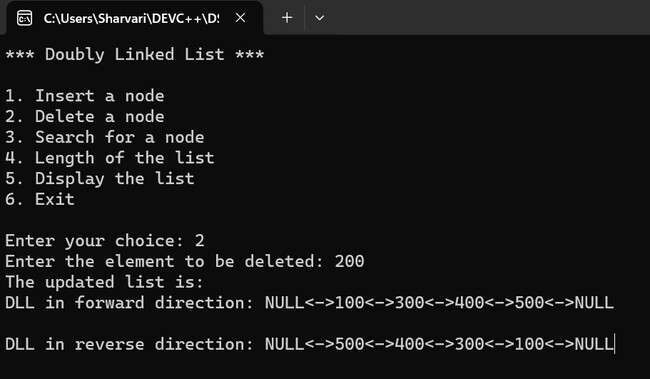
(Search)



(Length)



(Delete)



|  |  |
| --- | --- |
| **Program No:** | **23.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Polynomial Addition using Lists** |
| **Objective :** | **Unit-5: Linked Lists** |

**Date:** 28.10.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\*Name: Sharvari Birajdar  Roll No: 1409  Unit 5: Lists  Program: Addition of Polynomials \*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;    /\*----------1. Node Template--------\*/ **class** **PNode**  { **public:** **int** Coeff; **int** Exp;  PNode \*next;  };      /\*----------2.List Template-----------\*/    **class** **PList**  {  PNode \*head;    **public:** PList()  {  head = NULL; |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | }  **void** Insert(**int** c, **int** e); **void** **Display**(); **void** **Add**(PList P, PList Q);  **void** **Sub**(PList P, PList Q);  };      /\*--------------3.Functions----------------\*/ /\*-------------Insert Function---------------\*/ **void** PList :: Insert(**int** c,**int** e)  {  //Step-1: Make a new node t PNode \*t = **new** PNode;  t->Coeff = c; t->Exp = e;  t->next = NULL;    //Step-2 Special Case : add first node in the list **if**(head == NULL)  {  head = t;  }    **else**//Step-3 : traverse till the last node and attach t  {  PNode \*temp = head; //Creating a temporary pointer  PNode \*prev = NULL;    **while**(temp != NULL && temp->Exp > t->Exp)  {  prev = temp;  temp = temp->next;  }    **if**(temp==head) // 3a. Head Node Insertion  {  t->next = head;  head = t;  } |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **else** **if** (temp==NULL) //3b.Tail Node Deletion  {  prev->next = t;  }  **else** //3c.Insertion of any other nodes  {  prev->next = t;  t->next = temp;  }  }  }//end of Insert    /\*-------------Display Function---------------\*/ **void** PList :: Display()  {  cout << "The polynomial is : ";  PNode \*temp = head;    **while**(temp!= NULL)  {  **if**(temp->Coeff > **0**)  {  cout << "+";  }  cout << temp->Coeff << "x^" << temp->Exp << " "; temp= temp->next;  }//end of while      }//end of display    /\*-------------Add Function---------------\*/ **void** PList :: Add(PList x,PList y)  {  **this**-> head = NULL; //to eliminate extra terms    PNode \*P = x.head;  PNode \*Q= y.head;    **int** c,e; |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | **while**(P!=NULL && Q!=NULL)  {  **if**(P->Exp == Q->Exp)  {  c = P->Coeff + Q->Coeff;    **if**(c!=**0**)  {  e = P->Exp; //can be Q->Exp also  Insert(c,e);  }     1. = P->next; 2. = Q->next;     }//end of if(P==Q)      **else** **if**(P->Exp > Q->Exp)  {  c = P->Coeff; e = P->Exp;  Insert(c,e);    P = P->next;  }//end of if(P>Q)      **else** //(P->Exp < Q->Exp)  {  c = Q->Coeff; e = Q->Exp;  Insert(c,e);    Q = Q->next;  }//end of if(P<Q)    }//P&&Q    **while**(P!=NULL)  { |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | c = P->Coeff; e = P->Exp;  Insert(c,e);    P = P->next;  }    **while**(Q!=NULL)  {  c = Q->Coeff; e = Q->Exp;  Insert(c,e);    Q = Q->next;  }      }//end of add      /\*-------------Add Function---------------\*/ **void** PList :: Sub(PList x,PList y)  {  **this**-> head = NULL; //to eliminate extra terms      PNode \*P = x.head;  PNode \*Q= y.head;    **int** c,e;    **while**(P!=NULL && Q!=NULL)  {  **if**(P->Exp == Q->Exp)  {  c = P->Coeff - Q->Coeff;    **if**(c!=**0**)  {  e = P->Exp; //can be Q->Exp also  Insert(c,e); |

|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | }    //skip the term with 0 coefficient   1. = P->next; 2. = Q->next;     }//end of if(P==Q)      **else** **if**(P->Exp > Q->Exp)  {  c = P->Coeff; e = P->Exp;  Insert(c,e);    P = P->next;  }//end of if(P>Q)      **else** //(P->Exp < Q->Exp)  {  c = Q->Coeff; e = Q->Exp;  Insert(c,e);    Q = Q->next;  }//end of if(P<Q)    }//P&&Q    **while**(P!=NULL)  {  c = P->Coeff; e = P->Exp;  Insert(c,e);    P = P->next;  }    **while**(Q!=NULL)  { |

|  |  |
| --- | --- |
| 236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276 | c = Q->Coeff; e = Q->Exp;  Insert(c,e);    Q = Q->next;  }    }//end of sub    /\*-------------------4.Menu---------------\*/ **int** main()  {  **int** ch;    PList p1,p2,p3,p4;  **int** co,ex;    **while**(**1**)  {  system("cls");    cout << " \*\*\* Operations on Polynomials \*\*\***\n\n**"; cout << "1. Create 1st Polynomial**\n**"; cout << "2. Create 2nd Polynomial**\n**"; cout << "3. Display the 1st Polynomial**\n**"; cout << "4. Display the 2nd Polynomial**\n**"; cout << "5. Add the Polynomials**\n**"; cout << "6. Subtract the Polynomials**\n**"; cout << "7. Exit**\n\n**";    cout << "Enter your choice: "; cin>>ch;    **switch**(ch)  { **case** **1**:  cout << "Enter the coefficient: "; cin >> co;  cout << "Enter the exponent: "; cin >> ex;  p1.Insert(co,ex); |

|  |  |
| --- | --- |
| 277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317 | getch(); **break**;  **case** **2**:  cout << "Enter the coefficient: "; cin >> co;  cout << "Enter the exponent: "; cin >> ex;  p2.Insert(co,ex); getch(); **break**;  **case** **3**: p1.Display(); getch(); **break**;  **case** **4**: p2.Display(); getch(); **break**;  **case** **5**:  p3.Add(p1,p2); p3.Display(); getch(); **break**;  **case** **6**:  p4.Sub(p1,p2); p4.Display(); getch(); **break**;  **case** **7**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch(); |
| 318  319  320 | }//end of switch  }//end of while  }//end of main |

**Output:**

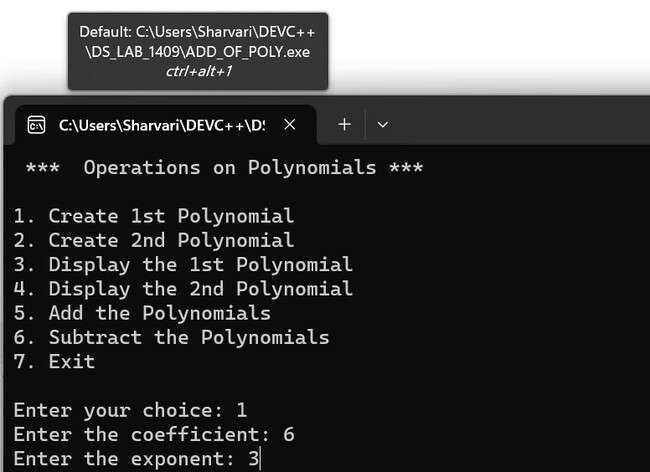
(

Create

1

st

polynomial)



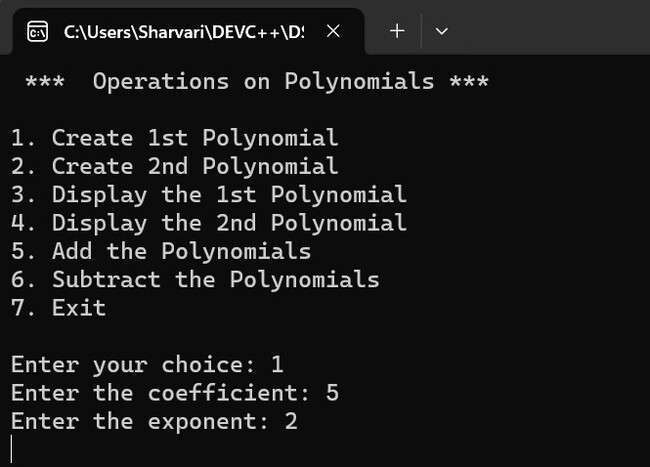
(

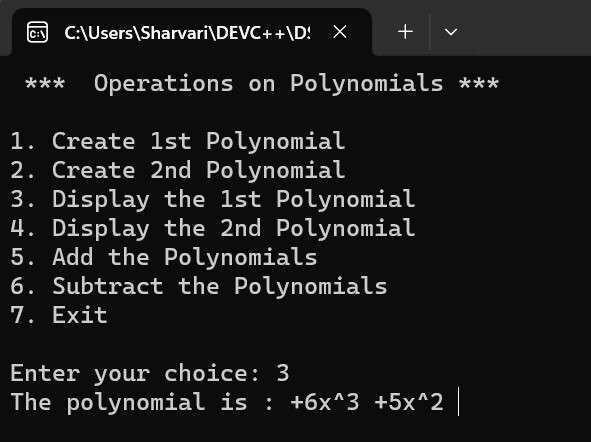
Display

1

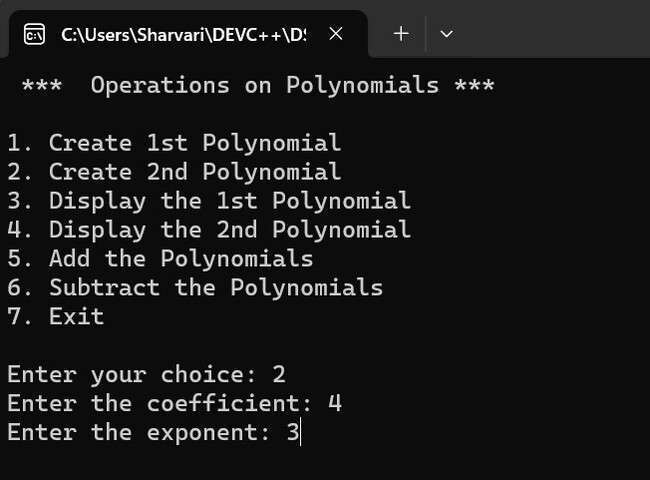
st

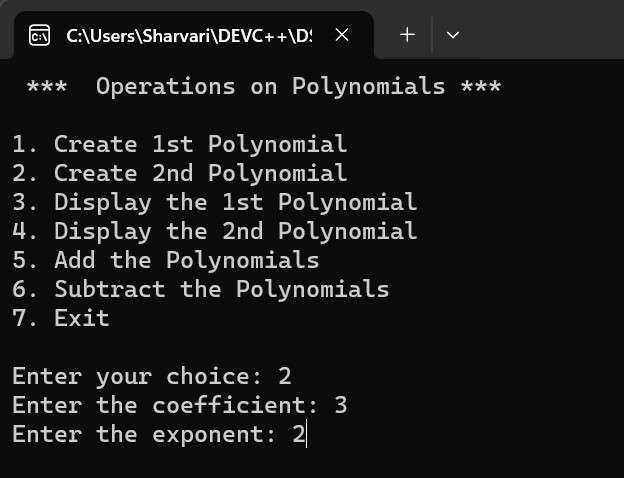
polynomial)



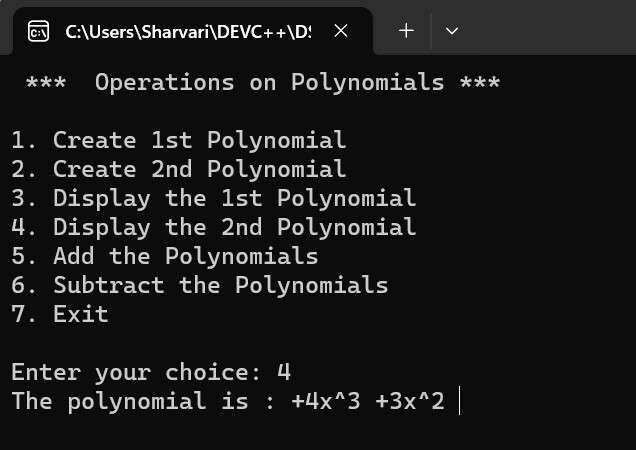


(Create 2nd Polynomial)

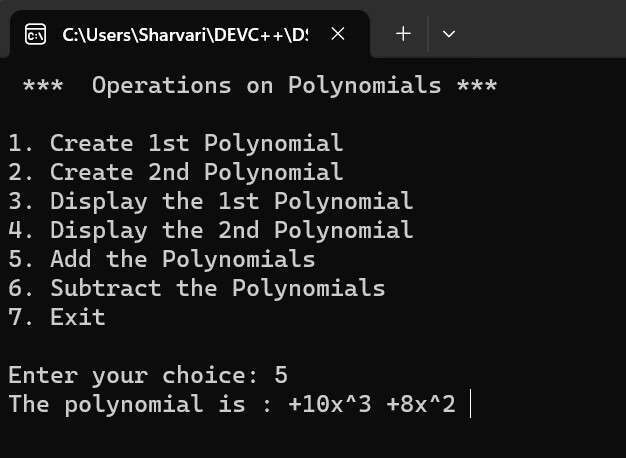




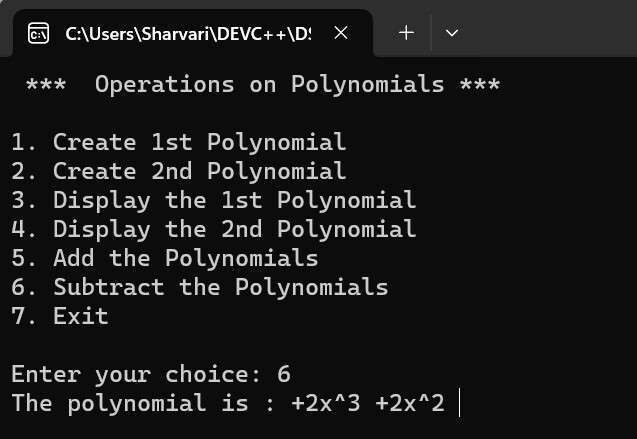
(Display 2nd Polynomial)



(Addition)



(Subtraction)



## Unit - 6 : Trees

|  |  |
| --- | --- |
| **Program No:** | **24.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Binary Search Trees** |
| **Objective :** | **Unit-6: Trees** |

**Date:** 05.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 6: Trees  Program: Binary Search Tree\*/    #include<iostream>  #include<conio.h>    **using** **namespace** std;  /\*----------1. Node Template ----------\*/    **class** **BSTNode**  { **public:**  **int** data;  BSTNode \*right;  BSTNode \*left;  };    /\*----------2.List Template--------\*/  **class** **BST**  {  BSTNode \*root;  **int** count;    **public:** BST()  {  root = NULL; count = **0**; |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | }    **void** Insert(**int** x); **void** **Search**(**int** x); **void** **Display**();  **void** **Inorder**(BSTNode \*p); **void** **Preorder**(BSTNode \*p); **void** **Postorder**(BSTNode \*p);  **void** **Count**(); **void** **FindMax**();  **void** **FindMin**();    };    /\*----------------3.Functions-------------\*/    /\*---------------Insert-------------\*/  **void** BST :: Insert(**int** x)  {  //Make a new Node  BSTNode \*t = **new** BSTNode;  t->data = x; t->right = NULL;  t->left = NULL;    //Special case - 1st node in the structure  **if**(root == NULL)  {  root = t; count++; **return**;  }    //Traverse through the BST  BSTNode \*temp = root;  BSTNode \*prev = NULL;    **while**(temp != NULL)  {  prev = temp; |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **if**(t->data < temp->data)  {  temp = temp->left;  }  **else** **if**(t->data > temp->data)  {  temp = temp->right;  } **else**  {  cout << "Duplicate values are not allowed"; getch(); **return**;  }  }//end of while      //Attach t to prev **if**(t->data < prev->data)  {  prev->left = t;  } **else** {  prev->right = t;  }    count++;    }//end of Insert    /\*-------------------Display--------------\*/  **void** BST :: Display()  {    cout << "Inorder Traversal: " << endl;  Inorder(root);  cout << endl;    cout << "Preorder Traversal: " << endl; Preorder(root); |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | cout << endl;    cout << "Postorder Traversal: " << endl;  Postorder(root);  cout << endl;    }//end of Display    /\*-------------------Inorder--------------\*/  **void** BST :: Inorder(BSTNode \*p)  {  **if**(p!= NULL)  {  Inorder(p->left); cout << p->data << " ";  Inorder(p->right);  }  }//end of Inorder    /\*-------------------Preorder--------------\*/  **void** BST :: Preorder(BSTNode \*p)  {  **if**(p!= NULL)  {  cout << p->data << " ";  Preorder(p->left);  Preorder(p->right);  }  }//end of Preorder    /\*-------------------Postorder--------------\*/ **void** BST :: Postorder(BSTNode \*p)  {  **if**(p!= NULL)  {  Postorder(p->left); Postorder(p->right);  cout << p->data << " ";  }  }//end of Postorder |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | /\*-------------------Count-----------------\*/  **void** BST :: Count()  {  cout << "The number of elements in the tree are: " << count;    }//end of Count    /\*------------------Search------------------\*/  **void** BST :: Search(**int** x)  {  BSTNode \*temp = root; **int** flag=**0**;    **while**(temp != NULL)  {  **if**(x < temp->data)  {  temp = temp->left;  }  **else** **if**(x > temp->data)  {  temp = temp->right;  }  **else** //Mil gayaaa  { flag=**1**; **break**;  }    }//end of while    **if**(flag==**1**)  {  cout << "The Element "<< x << " is found";  } **else**  {  cout << "The Element "<< x << " is not found";  } |

|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | }//end of Search    /\*------------------FindMax------------------\*/  **void** BST :: FindMax()  {  BSTNode \*temp = root;    **while**(temp->right != NULL)  {  temp = temp->right;  }    cout << "Maximum value in the BST: " << temp->data; }//end of FindMax    /\*------------------FindMin------------------\*/  **void** BST :: FindMin()  {  BSTNode \*temp = root;    **while**(temp->left != NULL)  {  temp = temp->left;  }    cout << "Minimum value in the BST: " << temp->data; }//end of FindMax    /\*------------------FindMin------------------\*/      /\*----------4.Menu--------\*/ **int** main()  {  **int** ch,num;  BST b;    **while**(**1**)  {  system("cls"); |

|  |  |
| --- | --- |
| 236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276 | cout <<" \*\*\*\* Binary Search tree \*\*\***\n\n**";  cout << "1. Insert the element**\n**"; cout << "2. Search for the element**\n**"; cout << "3. Display the tree**\n**"; cout << "4. Count**\n**"; cout << "5. Find the maximum element**\n**"; cout << "6. Find the minimum element**\n**"; cout << "7. Exit**\n**";    cout << "Enter your choice: "; cin >> ch;  **switch**(ch)  { **case** **1**:  cout << "Enter an element: "; cin>>num;  b.Insert(num); getch(); **break**;  **case** **2**:  cout << "Enter an element to be searched: "; cin>>num;  b.Search(num); getch(); **break**;  **case** **3**:  b.Display(); getch(); **break**;  **case** **4**:  b.Count(); getch(); **break**;  **case** **5**:  b.FindMax();  getch(); **break**; |

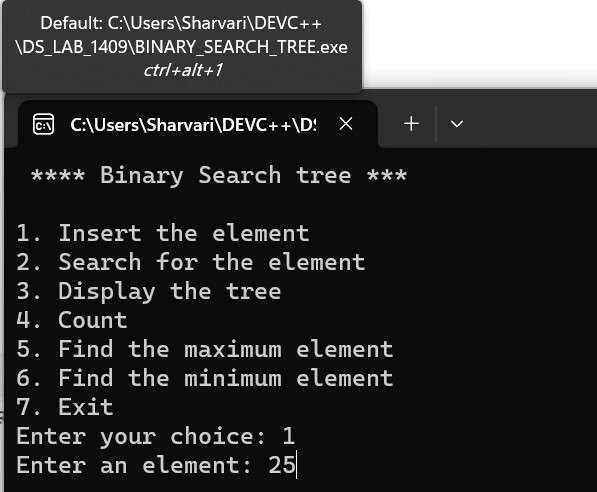
|  |  |
| --- | --- |
| 277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293 | **case** **6**:  b.FindMin(); getch(); **break**;  **case** **7**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch();  }//end of switch    }//end of while  }//end of main |

**Output:**

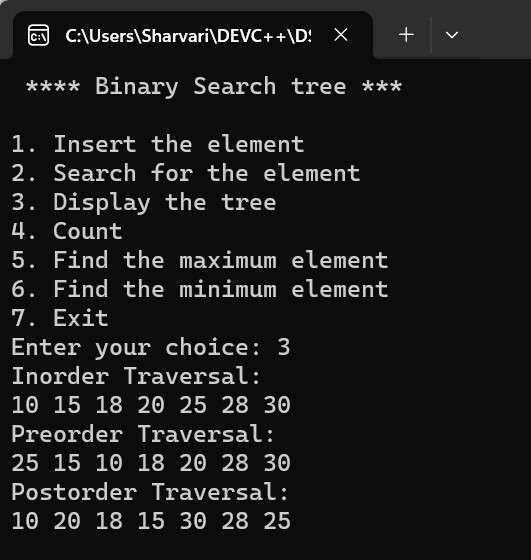
(

Insertion

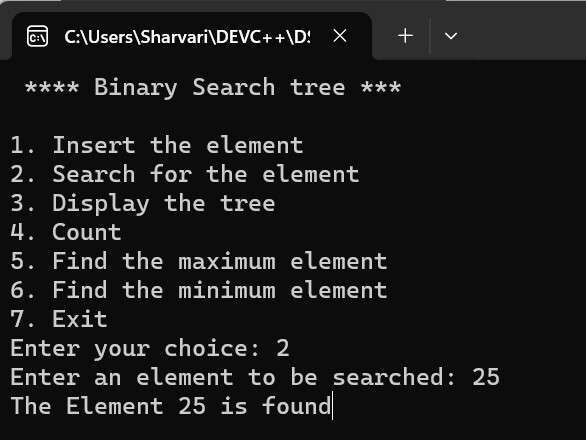
)

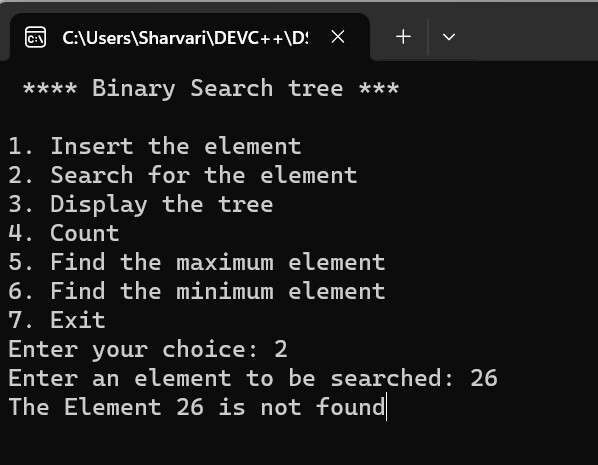


(Display)

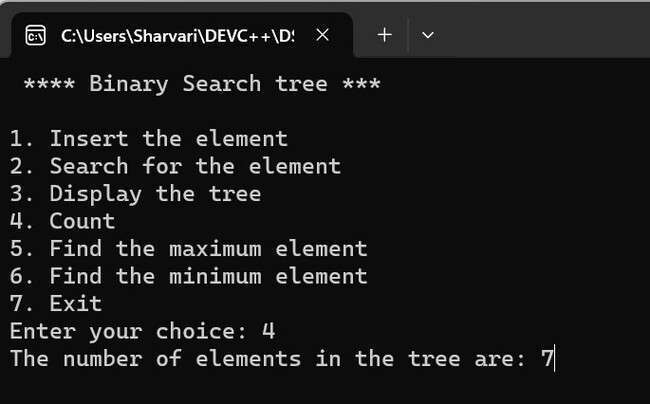


(Search)

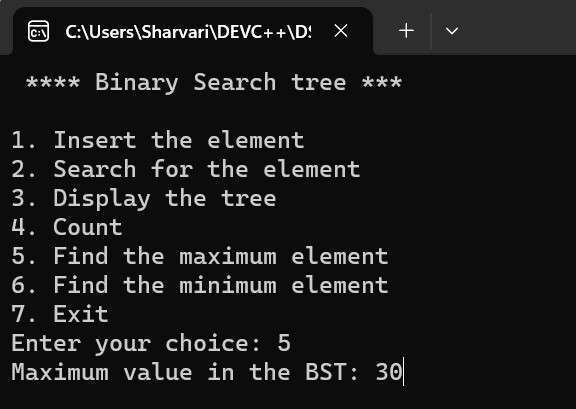




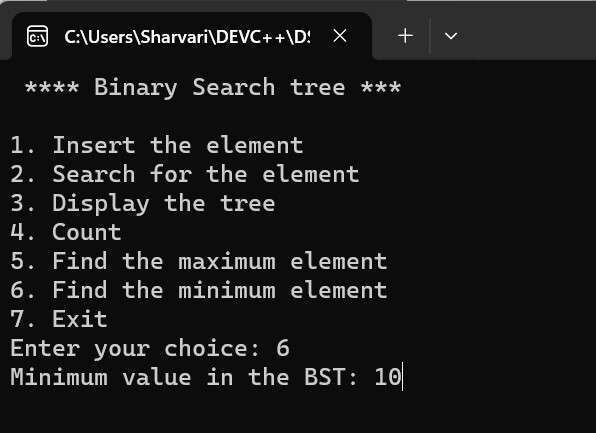
(Count)



(Maximum Element)



(Minimum Element)



|  |  |
| --- | --- |
| **Program No:** | **25.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Max Heaps** |
| **Objective :** | **Unit-6: Trees** |

**Date:** 11.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 6: Trees  Program: Max-Heap\*/    #include<iostream>  #include<conio.h>    #define MAX 20    **using** **namespace** std;    /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Max-Heap--------\*/ **class** **Heap**  {  **int** arr[MAX]; **int** n;    **public:** Heap()  { **int** i;  **for**(i=**0**;i<MAX;i++)  {  arr[i] = **0**;  } n=**0**; |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | }      **void** CreateHeap(); **void** **BuildHeap**(); **void** **InsertHeap**(**int** x); **void** **DeleteHeap**(); **void** **Display**(); **void** **reheapUp**(**int** i);  **void** **reheapDown**(**int** pos);    };    /\*----------------3.Functions------------------\*/    /\*---------------CreateHeap--------------\*/  **void** Heap::CreateHeap()  {  **int** i;    cout << "Enter the number of elements in the heap: "; cin >> n; //n declared in the Heap class    cout << "Enter the elements:**\n** ";    **for**(i=**0**; i<n; i++)  {  cout << "arr[" << i << "] = ";  cin >> arr[i];  }    BuildHeap();    }//end of CreateHeap      /\*----------------BuildHeap----------------\*/  **void** Heap::BuildHeap()  {  **int** i; |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **for**(i=(n/**2**)-**1**; i>=**0**; i--)  {  reheapDown(i);  }    }//end of BuildHeap    /\*----------------reheapDown--------------\*/ **void** Heap::reheapDown(**int** pos)  {  **int** i,val;    val = arr[pos];//Element to be floated down    **while**(pos <= (n/**2**)-**1**) // till last non-leaf position  {  i = (**2**\*pos) + **1**; //left child index    //Find the greater of left and right child **if**(arr[i] < arr[i+**1**])  {  i++; //maintain larger child in i }    //Compare val with larger child  **if**(val > arr[i])  {  **break**; //Nothing to be done  }    arr[pos] = arr[i]; //Shift larger child upwards      //Repeat the same steps on the ith index pos = i;    }//end of while    //Copy val to its correct location |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | arr[pos] = val;    }//end of reheapDown    /\*-------------------Display--------------\*/  **void** Heap::Display()  {  **if**(n == **0**)  {  cout << "Heap is empty";  **return**;  } **int** i;  **for**(i=**0**;i<n;i++)  {  cout << "arr[" << i << "] = " << arr[i] << endl;  }    }//end of Display    /\*------------------DeleteHeap--------------\*/  **void** Heap::DeleteHeap()  {  //Empty Heap  **if**(n==**0**)  {  cout << "Empty Heap!";  **return**;  }    **int** temp = arr[**0**]; //Copy the root element to temp    arr[**0**] = arr[n-**1**]; //Copy last element to root arr[n-**1**] = **0**; //Update last element to NIL n--; //Decrement total number of elements    reheapDown(**0**); //reheapDown on root    cout << "Element Deleted: " << temp << endl; |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | }//end of DeleteHeap    /\*------------------InsertHeap--------------\*/  **void** Heap::InsertHeap(**int** x)  {  arr[n] = x; //Insert x at nth position  n++; //Increment total number of elements    reheapUp(n-**1**); }//end of InsertHeap    /\*------------------reheapUp--------------\*/  **void** Heap::reheapUp(**int** i)  {  **int** val = arr[i]; //Copy element to be floated up in val    //arr[(i-1)/2] is the parent node    **while**(i>**0** && arr[(i-**1**)/**2**] <= val)  {  arr[i] = arr[(i-**1**)/**2**];  i = (i-**1**)/**2**; //Repeat the same steps on the parent index }//end of while    arr[i] = val; //Copy val to its final position    }//end of reheapUp    /\*------------------4.Menu--------------\*/ **int** main()  { **int** ch,num;  Heap h;    **while**(**1**)  { system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* Max-Heap\*\*\***\n\n**" | |   cout << ;    cout << "1.Create a Heap**\n**"; |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | "2.Insert an element in the Heap**\n**" | | | | | | "3.Delete Heap**\n**" | | ; | | | | "4.Display the Heap**\n**" | | | | ; | | "5.Exit**\n**" | ; | | | | "Enter your choice: " | | | ; |   cout << ; cout << cout << cout << cout <<  cin >> ch;      **switch**(ch)  {  **case** **1**:  h.CreateHeap();  getch();  **break**;  **case** **2**:  cout << "Enter the element to be inserted: "; cin >> num;  h.InsertHeap(num);  getch();  **break**;  **case** **3**:  h.DeleteHeap();  getch();  **break**;  **case** **4**:  h.Display(); getch();  **break**;  **case** **5**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch(); }//end of switlch  }//end of while |
| 236  237 | }//end of main |

**Output:**

(

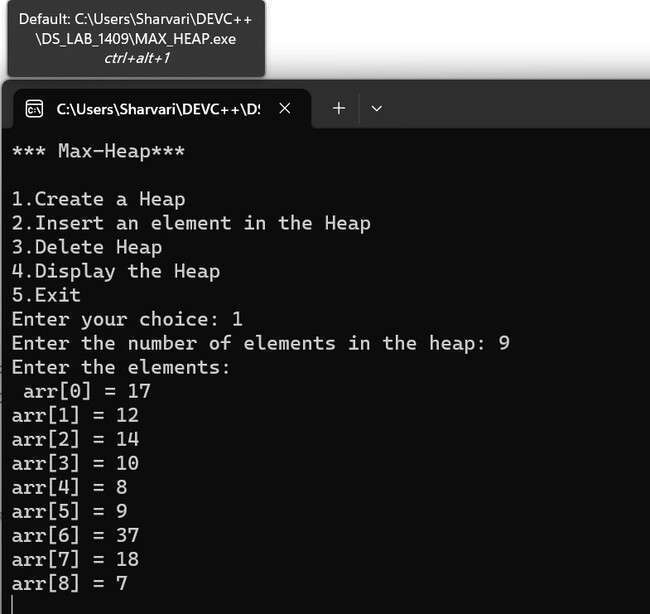
Create

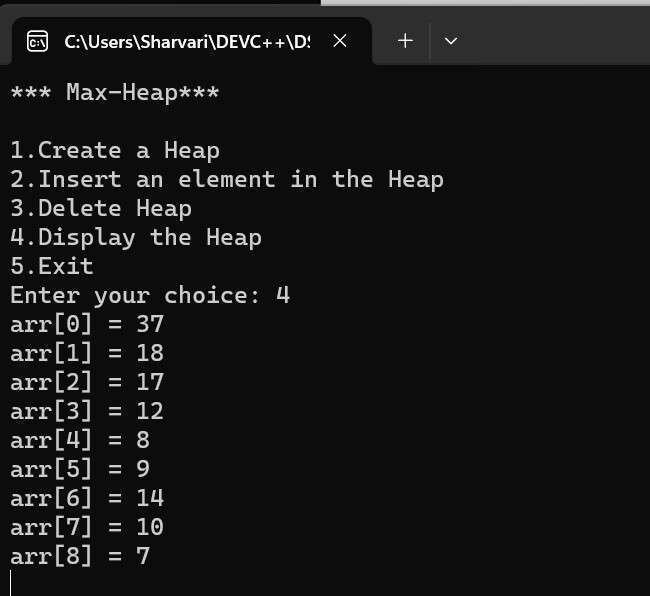
)

(

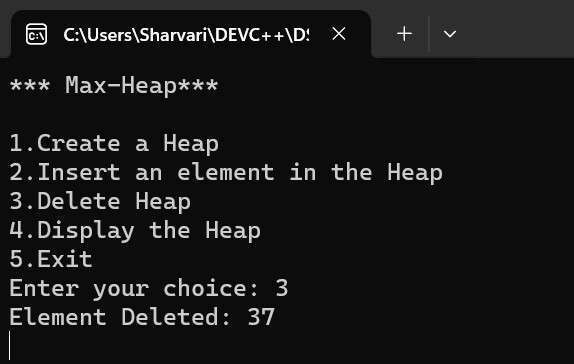
Display

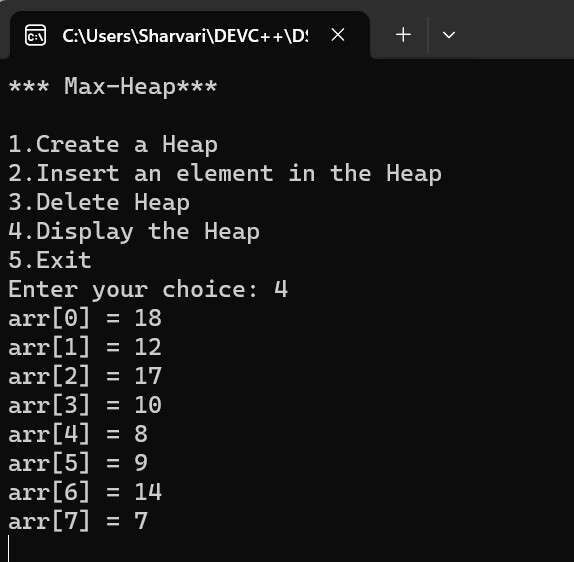
)



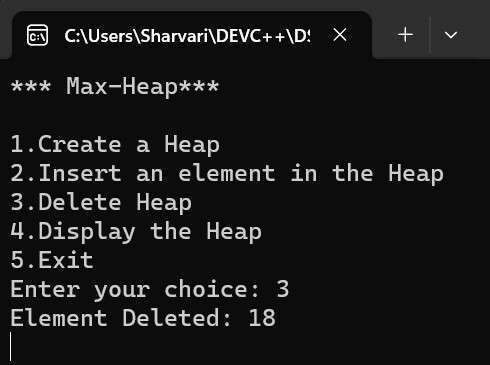


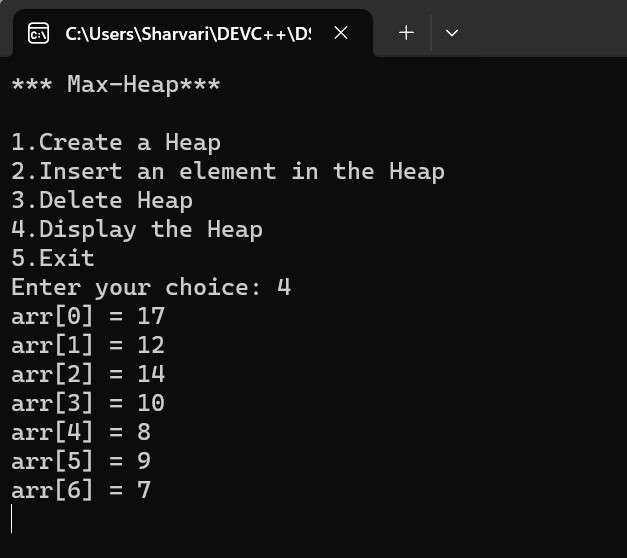
(Delete - 37)



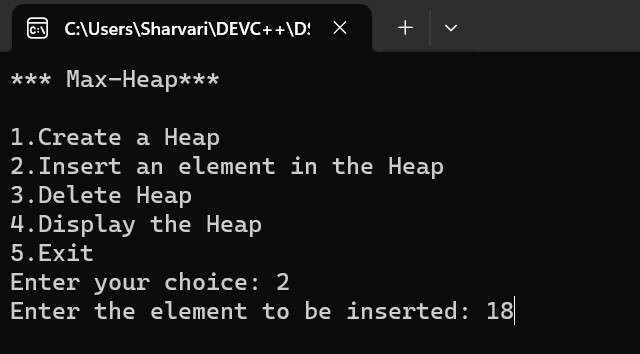


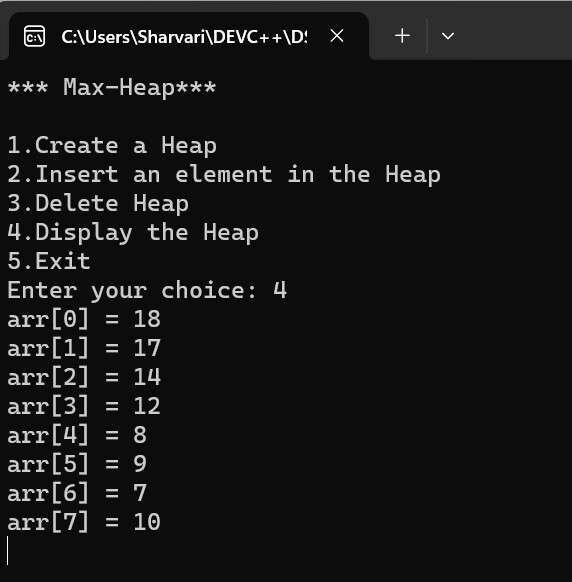
(Delete root - 18)





(Insert)





|  |  |
| --- | --- |
| **Program No:** | **26.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Min Heaps** |
| **Objective :** | **Unit-6: Trees** |

**Date:** 25.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 6: Trees  Program: Min-Heap\*/    #include<iostream>  #include<conio.h>    #define MAX 20    **using** **namespace** std;    /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Max-Heap--------\*/ **class** **Heap**  {  **int** arr[MAX]; **int** n;    **public:** Heap()  { **int** i;  **for**(i=**0**;i<MAX;i++)  {  arr[i] = **0**;  } n=**0**; |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | }      **void** **CreateHeap**(); **void** **BuildHeap**(); **void** **InsertHeap**(**int** x); **void** **DeleteHeap**(); **void** **Display**(); **void** **reheapUp**(**int** i);  **void** **reheapDown**(**int** pos);    };    /\*----------------3.Functions------------------\*/    /\*---------------CreateHeap--------------\*/  **void** Heap::CreateHeap()  {  **int** i;    cout << "Enter the number of elements in the heap: "; cin >> n; //n declared in the Heap class    cout << "Enter the elements:**\n** ";    **for**(i=**0**; i<n; i++)  {  cout << "arr[" << i << "] = ";  cin >> arr[i];  }    BuildHeap();    }//end of CreateHeap      /\*----------------BuildHeap----------------\*/  **void** Heap::BuildHeap()  {  **int** i; |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **for**(i=(n/**2**)-**1**; i>=**0**; i--)  {  reheapDown(i);  }    }//end of BuildHeap    /\*----------------reheapDown--------------\*/ **void** Heap::reheapDown(**int** pos)  {  **int** i,val;    val = arr[pos];//Element to be floated down    **while**(pos <= (n/**2**)-**1**) // till last non-leaf position  {  i = (**2**\*pos) + **1**; //left child index    //Find the smaller of left and right child **if**(i+**1**<n && arr[i] > arr[i+**1**])  {  i++; //maintain smaller child in i  }    //Compare val with smaller child  **if**(val <= arr[i])  {  **break**; //Nothing to be done  }    arr[pos] = arr[i]; //Shift smaller child upwards      //Repeat the same steps on the ith index pos = i;    }//end of while    //Copy val to its correct location |

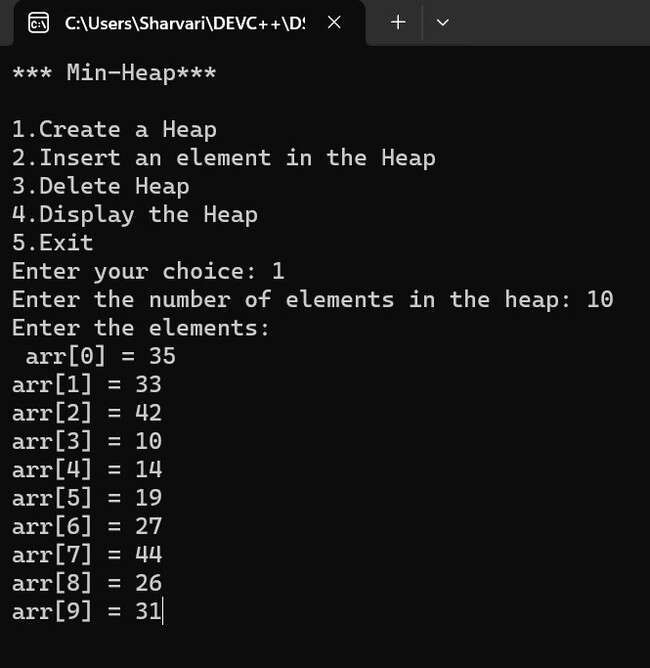
|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | arr[pos] = val;    }//end of reheapDown    /\*-------------------Display--------------\*/  **void** Heap::Display()  {  **if**(n == **0**)  {  cout << "Heap is empty";  **return**;  } **int** i;  **for**(i=**0**;i<n;i++)  {  cout << "arr[" << i << "] = " << arr[i] << endl;  }    }//end of Display    /\*------------------DeleteHeap--------------\*/  **void** Heap::DeleteHeap()  {  //Empty Heap  **if**(n==**0**)  {  cout << "Empty Heap!";  **return**;  }    **int** temp = arr[**0**]; //Copy the root element to temp    arr[**0**] = arr[n-**1**]; //Copy last element to root arr[n-**1**] = **0**; //Update last element to NIL n--; //Decrement total number of elements    reheapDown(**0**); //reheapDown on root    cout << "Element Deleted: " << temp << endl; |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | }//end of DeleteHeap    /\*------------------InsertHeap--------------\*/  **void** Heap::InsertHeap(**int** x)  {  arr[n] = x; //Insert x at nth position  n++; //Increment total number of elements    reheapUp(n-**1**); }//end of InsertHeap    /\*------------------reheapUp--------------\*/  **void** Heap::reheapUp(**int** i)  {  **int** val = arr[i]; //Copy element to be floated up in val    //arr[(i-1)/2] is the parent node    **while**(i>**0** && arr[(i-**1**)/**2**] > val)  {  arr[i] = arr[(i-**1**)/**2**];  i = (i-**1**)/**2**; //Repeat the same steps on the parent index }//end of while    arr[i] = val; //Copy val to its final position    }//end of reheapUp    /\*------------------4.Menu--------------\*/ **int** main()  {  **int** ch,num;  Heap h;    **while**(**1**)  { system(   |  |  | | --- | --- | | "cls" | ); | | "\*\*\* Min-Heap\*\*\***\n\n**" | |   cout << ;    cout << "1.Create a Heap**\n**"; |

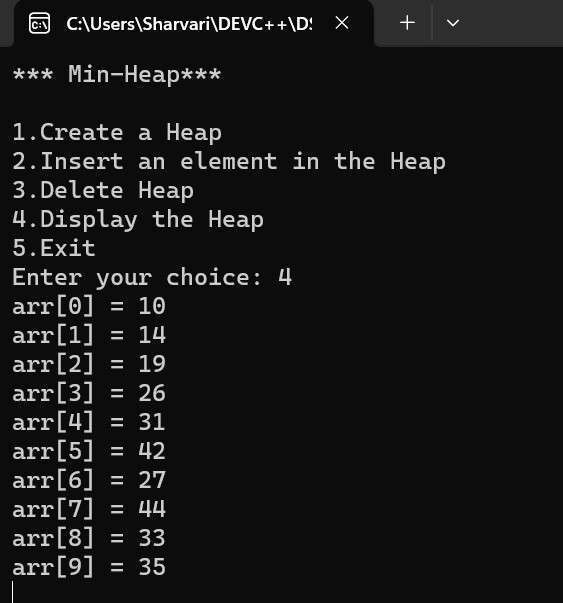
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | "2.Insert an element in the Heap**\n**" | | | | | | "3.Delete Heap**\n**" | | ; | | | | "4.Display the Heap**\n**" | | | | ; | | "5.Exit**\n**" | ; | | | | "Enter your choice: " | | | ; |   cout << ; cout << cout << cout << cout <<  cin >> ch;      **switch**(ch)  {  **case** **1**:  h.CreateHeap();  getch();  **break**;  **case** **2**:  cout << "Enter the element to be inserted: "; cin >> num;  h.InsertHeap(num);  getch();  **break**;  **case** **3**:  h.DeleteHeap();  getch();  **break**;  **case** **4**:  h.Display(); getch();  **break**;  **case** **5**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch(); }//end of switch  }//end of while |
| 236  237 | }//end of main |

**Output:**

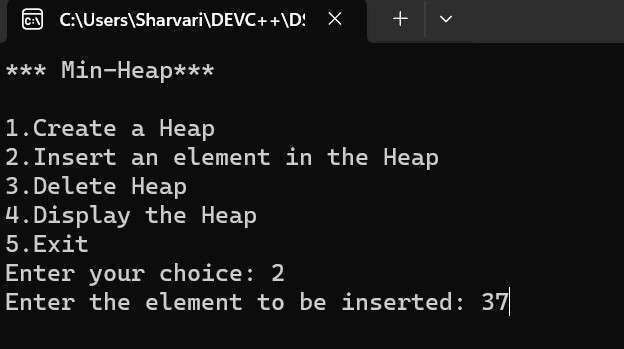
(Create a heap)

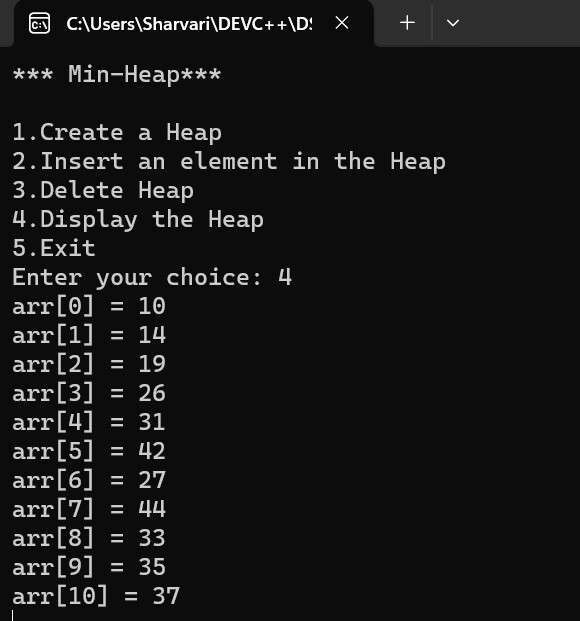


(Display)

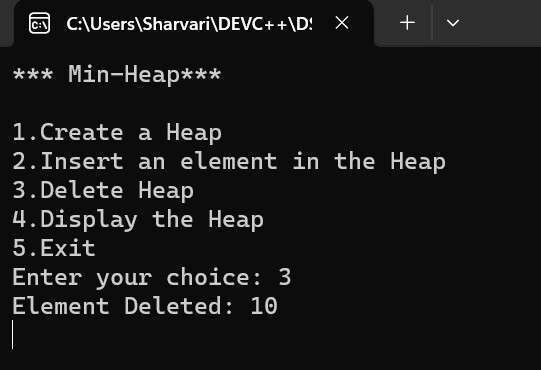


(Insert)





(Delete heap - root 10 is deleted)



**Unit-7:Graphs**

|  |  |
| --- | --- |
| **Program No:** | **27.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Representation of Graphs using Adjacency Matrix** |
| **Objective :** | **Unit-7: Graphs** |

**Date:** 07.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 7: Graphs  Program: Adjacency Matrix\*/    #include<iostream>  #include<conio.h>    #define MAX 10  **using** **namespace** std;    /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Graph--------\*/ **class** **Graph**  {  **int** adj[MAX][MAX];  **int** n,e;    **public:** Graph()  {  **int** i,j;    **for**(i=**0**; i<MAX; i++)  {  **for**(j=**0**; j<MAX; j++) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | {  adj[i][j] = **0**;  }  } n = **0**; e = **0**;  }    **void** CreateGraph(); **void** **Display**();  };    /\*------------3.Functions-------------\*/    /\*---------------CreateGraph-------------\*/  **void** Graph :: CreateGraph()  { cout << "Enter the number of vertices: "; cin >> n; //n is the declared in the Graph Class    cout << "Enter the number of edges: "; cin >> e; //e is declared in the Graph Class      **int** source,dest; **int** ne=**1**;    **while**(ne<=e)  {   |  |  | | --- | --- | | "Enter Edge " | << | | "Enter the source vertex: " | |   cout << ne << endl;  cout << ;  cin >> source;    cout << "Enter the destination vertex: "; cin >> dest;    adj[source][dest] = **1**; adj[dest][source] = **1**;    ne++; |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111 | }    }//end of CreateGraph    /\*-------------------Display----------------\*/  **void** Graph :: Display()  {  **int** i,j;  cout << "The Adjacency Matrix of the graph is: " << endl; **for**(i=**0**;i<n;i++)  {  **for**(j=**0**;j<n;j++)  {  cout << adj[i][j] << " ";  }  cout<<endl;  }    }//end of Display    /\*------------------4.Menu--------------\*/ **int** main()  {  **int** ch;  Graph g;    **while**(**1**)  {  system("cls");     |  |  |  |  | | --- | --- | --- | --- | | "\*\*\* Graph - Adjacency Matrix \*\*\***\n\n**" | | | | | "1.Create a graph**\n**" | | ; | | | "2.Display the graph**\n**" | | | ; | | "3.Exit**\n**" | ; | |   cout << ; cout << cout << cout <<    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  { **case** **1**: |

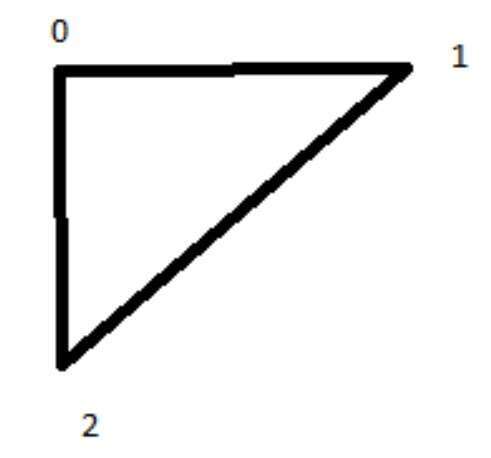
|  |  |
| --- | --- |
| 112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127 | g.CreateGraph();  getch();  **break**;  **case** **2**:  g.Display(); getch(); **break**; **case** **3**:  exit(**1**); **default:** cout << "Incorrect Choice!"; getch(); }//end of switch  }//end of while  }//end of main |

**Output:**

(

Graph

)

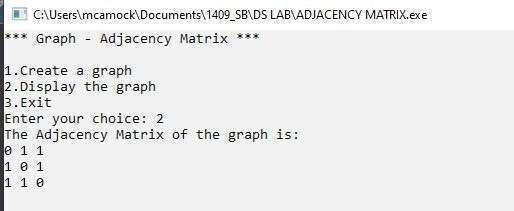
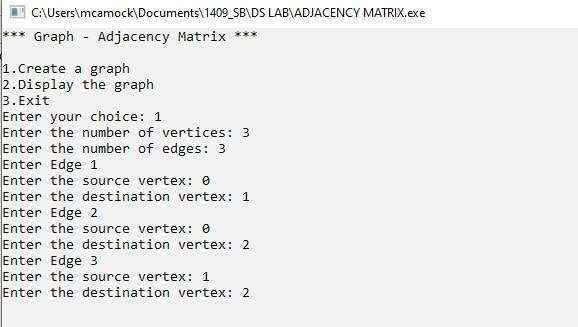


(Create a graph)

(

Display

)



|  |  |
| --- | --- |
| **Program No:** | **28.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Perform Breadth-First Traversal on a Graph using Queues** |
| **Objective :** | **Unit-7: Graphs** |

**Date:** 07.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 7: Graphs  Program: Breadth First Traversal\*/    #include<iostream>  #include<conio.h>    #define MAX 10  **using** **namespace** std;    /\*---------- Queue Template ----------\*/  **class** **Queue**  { **int** a[MAX];  **int** front,rear;    **public:** Queue()  {  front = -**1**; rear = -**1**;  }    **void** Enqueue(**int** x); **int** **Dequeue**(); **int** **Empty**();  }; |

|  |  |
| --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **int** Queue :: Empty()  {  **if**(front == -**1**)  {  **return** **1**;  } **else** {  **return** **0**;  }  }    **void** Queue :: Enqueue(**int** x)  { rear++;  a[rear] = x;    **if**(front == -**1**)  {  front++;  }  }    **int** Queue :: Dequeue()  {  **if**(Empty())  {  cout << "Queue Underflow!";  **return** -**1**;  }    **int** temp = a[front];    **if**(front == rear)  {  front = rear = -**1**;  } **else** {  front++;  } |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | **return** temp;    }    /\*---------- Graph Template ----------\*/  /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Graph--------\*/ **class** **Graph**  {  **int** adj[MAX][MAX];  **int** visited[MAX]; **int** n,e;    **public:** Graph()  {  **int** i,j;    **for**(i=**0**; i<MAX; i++)  {  **for**(j=**0**; j<MAX; j++)  {  adj[i][j] = **0**;  }  } n = **0**; e = **0**;  }    **void** CreateGraph(); **void** **Display**(); **void** **BFT**(**int** x);  };    /\*------------3.Functions-------------\*/    /\*---------------CreateGraph-------------\*/  **void** Graph :: CreateGraph() { |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | cout << "Enter the number of vertices: "; cin >> n; //n is the declared in the Graph Class    cout << "Enter the number of edges: "; cin >> e; //e is declared in the Graph Class      **int** source,dest; **int** ne=**1**;    **while**(ne<=e)  {   |  |  | | --- | --- | | "Enter Edge " | << | | "Enter the source vertex: " | |   cout << ne << endl;  cout << ;  cin >> source;    cout << "Enter the destination vertex: "; cin >> dest;    adj[source][dest] = **1**; adj[dest][source] = **1**;    ne++;  }    }//end of CreateGraph    /\*-------------------Display----------------\*/  **void** Graph :: Display()  {  **int** i,j;  cout << "The Adjacency Matrix of the graph is: " << endl; **for**(i=**0**;i<n;i++)  {  **for**(j=**0**;j<n;j++)  {  cout << adj[i][j] << " ";  }  cout<<endl;  } |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | }//end of Display    **void** Graph :: BFT(**int** x)  {    Queue q;  **int** i;    **for**(i=**0**; i<MAX; i++)  {  visited[i] = **0**;  }    //Update the starting vertex visited[x] = **1**;  q.Enqueue(x);    cout << "Order of Traversal: ";    **while**(!q.Empty())  {  x = q.Dequeue();    cout << x << " ";    **for**(i=**0**; i<n; i++)  {  **if**(adj[x][i] == **1** && visited[i] == **0**) //Neighbour & Visited Status  {  visited[i] = **1**;  q.Enqueue(i);  }  }//end of i  }//end of while      }//end of BFT    /\*------------------4.Menu--------------\*/ **int** main() |

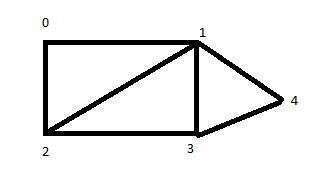
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | { **int** ch,num;  Graph g;    **while**(**1**)  {  system("cls");     |  |  |  |  |  | | --- | --- | --- | --- | --- | | "\*\*\* Graph - Breadth First Traversal \*\*\***\n\n**" | | | | | | "1.Create a graph**\n**" | | ; | | | | "2.Display the graph**\n**" | | | ; | | | "3.Breadth First Traversal**\n**" | | | | ; | | "4.Exit**\n**" | ; | | |   cout << ; cout << cout << cout << cout <<    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  { **case** **1**:  g.CreateGraph();  getch();  **break**;  **case** **2**:  g.Display(); getch(); **break**; **case** **3**:  cout << "Enter the starting vertex: "; cin >> num;    g.BFT(num); getch(); **break**; **case** **4**:  exit(**1**); **default:** cout << "Incorrect Choice!"; getch();  }//end of switch |
| 236  237  238 | }//end of while  }//end of main |

**Output:**

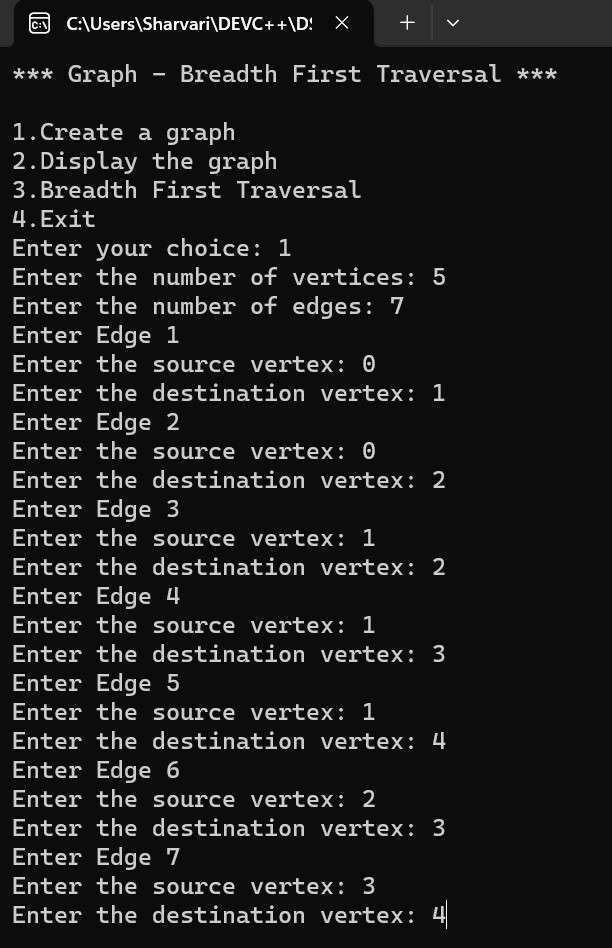
(

Graph

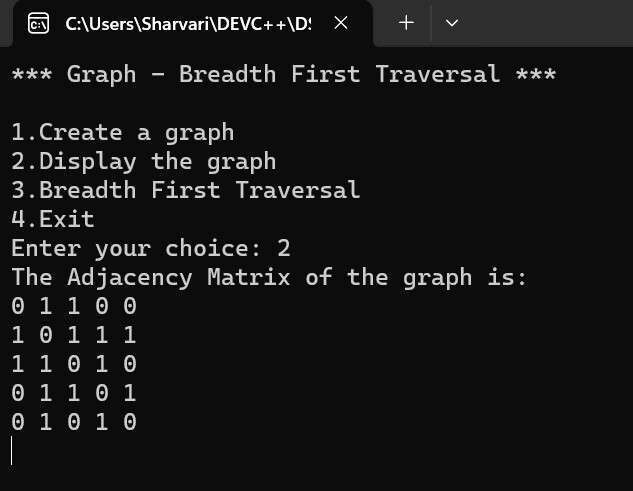
)



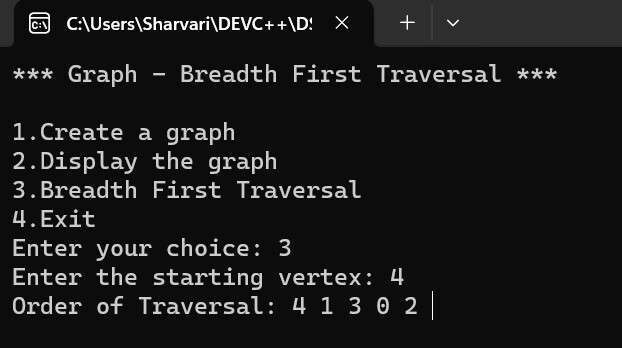
(Create a graph)



(Display)



### (Breadth First Traversal)



|  |  |
| --- | --- |
| **Program No:** | **29.** |
| **Roll No :** | **1409** |
| **Title of Program :** | **Find MST using Kruskal’s Algorithm** |
| **Objective :** | **Unit-7: Graphs** |

**Date:** 09.12.2023

**Source Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\* Name: Sharvari Birajdar  Roll no: 1409  Unit 7: Graphs  Program: Kruskal's Algorithm for MST\*/    #include<iostream>  #include<conio.h>    #define MAX 10 #define INF 999  **using** **namespace** std;    /\*----------1. Node Template -Not required----------\*/    /\*----------2.Array Based Template for Graph--------\*/ **class** **Graph**  {  **int** adj[MAX][MAX]; **int** n,e;  **int** parent[MAX];    **public:** Graph()  {  **int** i,j;    **for**(i=**0**; i<MAX; i++)  { |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | **for**(j=**0**; j<MAX; j++)  {  adj[i][j] = INF;  }  }  **for**(i=**0**;i<MAX;i++)  {  parent[i] = -**1**;  } n = **0**; e = **0**;  }    **void** CreateGraph(); **void** **Display**(); **void** **Kruskal**(); **int** **Find**(**int** i); **int** **Union**(**int** i,**int** j);  };    /\*------------3.Functions-------------\*/    /\*---------------CreateGraph-------------\*/  **void** Graph :: CreateGraph()  { cout << "Enter the number of vertices: "; cin >> n; //n is the declared in the Graph Class    cout << "Enter the number of edges: "; cin >> e; //e is declared in the Graph Class    **int** source, dest, weight; **int** ne=**1**;    **while**(ne<=e)  {   |  |  | | --- | --- | | "Enter Edge " | << | | "Enter the source vertex: " | |   cout << ne << endl;  cout << ;  cin >> source; |

|  |  |
| --- | --- |
| 72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112 | cout << "Enter the destination vertex: "; cin >> dest;    cout << "Enter weight: "; cin >> weight;    adj[source][dest] = weight; adj[dest][source] = weight;    ne++;  }    }//end of CreateGraph    /\*-------------------Display----------------\*/  **void** Graph :: Display()  {  **int** i,j;  cout << "The Adjacency Matrix of the graph is: " << endl; **for**(i=**0**;i<n;i++)  {  **for**(j=**0**;j<n;j++)  {  cout << adj[i][j] << " ";  }  cout<<endl;  }    }//end of Display    /\*-----------------Kruskal----------------\*/  **void** Graph :: Kruskal()  { **int** minimum,i,j; **int** ne=**1**;//ne- loop  **int** a,b,u,v;  **int** mincost=**0**;      **while**(ne < n) |

|  |  |
| --- | --- |
| 113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153 | {  //Find the minimum edge  **for**(i=**0**,minimum=**999**; i<n; i++)  {  **for**(j=**0**; j<n; j++)  {  **if**(adj[i][j] < minimum)  {  minimum = adj[i][j];  a = u = i; b = v = j;  }//end of j  }  }//end of i    u = Find(u); v = Find(v);    **if**(Union(u,v))  {  cout << "Edge: " << ne++ << ": (" << a << " , " << b << ") = " << adj[a][b] << endl; mincost += minimum;  }    //To ensure Edge is not selected again adj[a][b] = adj[b][a] = INF;    }//end of while    cout << "MST Total Cost is: " << mincost;    }//end of Kruskal    /\*------------------Find------------------\*/  **int** Graph :: Find(**int** i)  {  **while**(parent[i]!=-**1**)  {  i = parent[i];  } |

|  |  |
| --- | --- |
| 154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194 | **return** i;  }    /\*----------------Union--------------------\*/ **int** Graph :: Union(**int** i,**int** j)  {  **if**(i!=j)  { parent[j] = i; **return** **1**;  }  }    /\*------------------4.Menu--------------\*/ **int** main()  {  **int** ch;  Graph g;    **while**(**1**)  {  system("cls");    cout << "\*\*\* Graph - Minimum Spanning Tree \*\*\***\n\n**";    cout << "1.Create a graph**\n**"; cout << "2.Display the graph**\n**"; cout << "3.MST-Kruskal's Algorithm'**\n**"; cout << "4.Exit**\n**";    cout << "Enter your choice: "; cin >> ch;    **switch**(ch)  { **case** **1**:  g.CreateGraph();  getch();  **break**; |

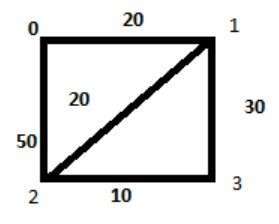
|  |  |
| --- | --- |
| 195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213 | **case** **2**:  g.Display(); getch();  **break**;  **case** **3**:  g.Kruskal(); getch();  **break**;  **case** **4**:  exit(**1**);  **default:** cout << "Incorrect Choice!"; getch(); }//end of switch  }//end of while  }//end of main |

**Output:**

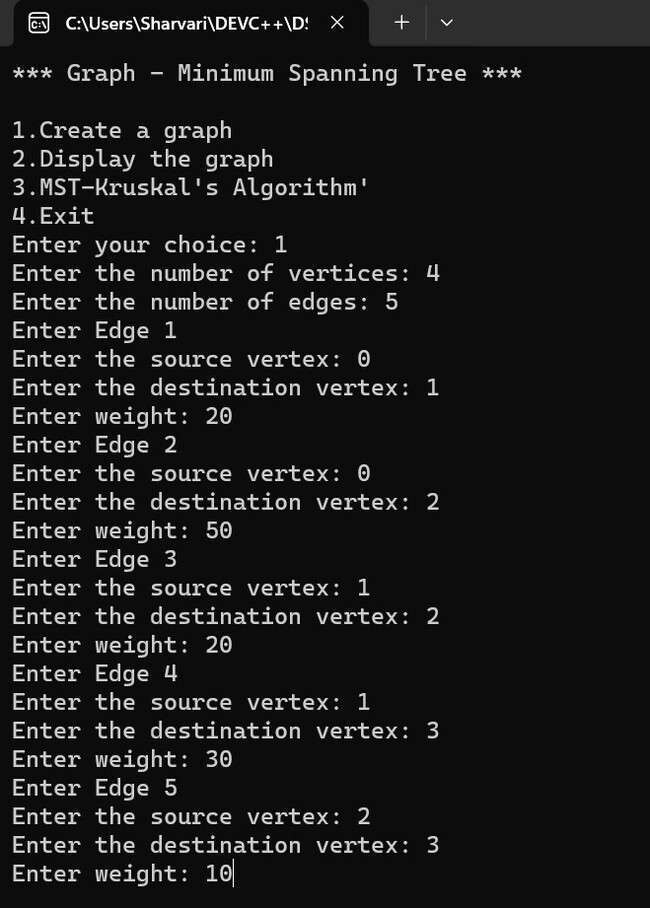
(

Graph

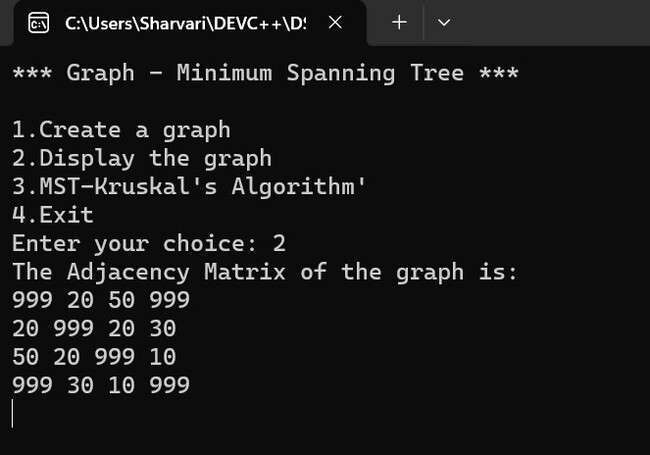
)



(Create a graph)



(Display the adjacency matrix)



### (MST-Kruskal’s Algorithm)

