**Write a program that implements the following sorting methods to sort a given list of**

**integers in ascending order**

**i) Bubble sort**

#include<stdio.h>

void Bubble\_sort(int a[100],int n)

{

     for(int i=0;i<n-1;i++)

     {

             for(int j=0;j<n-i-1;j++)

             {

                     if(a[j]>a[j+1])

                     {

                             int t=a[j];

                             a[j]=a[j+1];

                             a[j+1]=t;

                     }

             }

     }

}

int main()

{

     int n,a[30];

     scanf("%d",&n);

     for(int i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     Bubble\_sort(a,n);

     printf("Elements after sorting\n");

     for(int i=0;i<n;i++)

     {

             printf("%d\n",a[i]);

     }

}

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ vim bubble\_sort.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc bubble\_sort.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

6

4 6 23 32 2 22

Elements after sorting

2

4

6

22

23

32

**ii) Selection sort**

#include<stdio.h>

void Selection\_sort(int a[100],int n)

{

     int i,j,t,min;

     for(i=0;i<n;i++)

     {

             min=i;

             for(j=i+1;j<n;j++)

             {

                     if(a[j]<a[min])

                     {

                             min=j;

                     }

             }

             t=a[min];

             a[min]=a[i];

             a[i]=t;

     }

}

int main()

{

     int n,a[30];

     scanf("%d",&n);

     for(int i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     Selection\_sort(a,n);

     printf("Elements after sorting\n");

     for(int i=0;i<n;i++)

     {

             printf("%d\n",a[i]);

     }

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ vim selection\_sort.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc selection\_sort.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

6

23 24 12 11 32 9

Elements after sorting

9

11

12

23

24

32

**iii) Insertion sort**

#include<stdio.h>

void Insertion\_sort(int a[100],int n)

{

     for(int i=1;i<n;i++)

     {

             int index = a[i];

             int j=i;

             while((j>0) && (a[j-1]>index))

             {

                     a[j]=a[j-1];

                     j=j-1;

             }

             a[j] = index;

     }

}

int main()

{

     int n,a[30];

     scanf("%d",&n);

     for(int i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     Insertion\_sort(a,n);

     printf("Elements after sorting\n");

     for(int i=0;i<n;i++)

     {

             printf("%d\n",a[i]);

     }

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ vim insertion\_sort.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc insertion\_sort.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

4

40 20 30 10

Elements after sorting

10

20

30

40

**Write a C Program for Linear Search using non-recursion**

            #include<stdio.h>

int linear\_search(int a[50],int n,int key)

{

     int i,pos=-1;

     for(i=0;i<n;i++)

     {

             if(a[i]==key)

             {

                     pos=i;

                     break;

             }

     }

     return pos;

}

int main()

{

     int i,pos,n,key,a[50];

     scanf("%d",&n);

     for(i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     scanf("%d",&key);

     pos=linear\_search(a,n,key);

     if(pos==-1)

             printf("element not present\n");

     else

             printf("element present at %d\n",pos+1);

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc linearSearch.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

5

1 3 4 6 7

6

element present at 4

**Write a C Program for Linear Search using recursion**

            //Linear Search using Recursion

#include<stdio.h>

int linear\_search(int a[50],int n,int key)

{

     static int i;

     if(a[i] == key)

     {

             return i;

     }

     else if(i<n)

     {

             i++;

             linear\_search(a,n,key);

     }

     else

     {

             return -1;

     }

}

int main()

{

     int i,pos,n,key,a[50];

     scanf("%d",&n);

     for(i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     scanf("%d",&key);

     pos=linear\_search(a,n,key);

     if(pos==-1)

             printf("element not present\n");

     else

             printf("element present at %d\n",pos+1);

}

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc linearSearch.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

6

2 6 1 3 7 9

9

element present at 6

**Write a C Program for Binary Search using non-recursion**

#include<stdio.h>

int binarysearch(int first,int last,int a[50],int key)

{

     int pos=-1,mid;

     while(first<=last)

     {

             mid=(first+last)/2;

             if(a[mid]==key)

             {

                     pos=mid;

                     break;

             }

             else if(a[mid]<key)

             {

                     first=mid+1;

             }

             else

             {

                     last=mid-1;

             }

     }

            return pos;

}

int main()

{

     int i,pos,n,key,a[50];

     scanf("%d",&n);

     for(i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     scanf("%d",&key);

     pos=binarysearch(0,n-1,a,key);

     if(pos==-1)

             printf("element not present\n");

     else

             printf("element present at %d\n",pos+1);

}

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc binarySearch.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

6

2 3 4 5 6 7

3

element present at 2

**Write a C Program for Binary Search using recursion**

            #include<stdio.h>

int binarysearch(int first,int last,int a[50],int key)

{

     int mid;

     if(first<=last)

     {

             mid=(first+last)/2;

             if(a[mid] == key)

             {

                     return mid;

             }

             else if(a[mid]<key)

             {

                     binarysearch(mid+1,last,a,key);

}

             else

             {

                     binarysearch(first,mid-1,a,key);

             }

     }

            else

     return -1;

}

int main()

{

     int i,pos,n,key,a[50];

     scanf("%d",&n);

     for(i=0;i<n;i++)

     {

             scanf("%d",&a[i]);

     }

     scanf("%d",&key);

     pos=binarysearch(0,n-1,a,key);

     if(pos==-1)

             printf("element not present\n");

     else

             printf("element present at %d\n",pos+1);

}

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc recursiveBinary.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

6

2 3 5 6 7 8

5

element present at 3

**Write a program that uses functions to perform the following operations on singly linked list:**

**i) Creation**

**ii)Insertion**

**iii)Deletion**

**iv)Traversal**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*link;

};

struct node\*head = NULL,\*tail=NULL,\*cur,\*next,\*prev;

void create()

{

    int n;

    printf("Enter number of nodes we need to create\n");

    scanf("%d",&n);

    for(int i=1;i<n;i++)

    {

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("current node data\n");

    scanf("%d",&(cur->data));

    cur->link = NULL;

    if(head==NULL)

    {

    head=tail=cur;

    }

    else

    {

    tail->link=cur;

    tail=cur;

    }

    }

}

void insert\_at\_begin()

{

    cur = (struct node\*)malloc(sizeof(struct node));

    printf("Enter data\n");

    scanf("%d",&(cur->data));

    printf("%d",(cur->data));

    cur->link = head;

    head=cur;

}

void insert\_at\_end()

{

    cur = (struct node\*)malloc(sizeof(struct node));

    printf("enter data\n");

    scanf("%d",&(cur->data));

    cur->link=NULL;

    tail->link=cur;

    tail=cur;

}

void insert\_at\_pos()

{

    int pos,c=1;

    cur = (struct node\*)malloc(sizeof(struct node));

    printf("Enter data\n");

    scanf("%d",&(cur->data));

    printf("Enter position\n");

    scanf("%d",&pos);

    next=head;

    while(c<pos)

    {

    prev=next;

    next=next->link;

    c++;

    }

    prev->link=cur;

    cur->link=next;

}

void insert\_before()

{

    int value;

    cur = (struct node\*)malloc(sizeof(struct node));

    printf("Enter data\n");

    scanf("%d",&(cur->data));

    printf("Enter value\n");

    scanf("%d",&value);

    next=head;

    while(next->data != value && next!= NULL)

    {

    prev=next;

    next=next->link;

    }

    prev->link=cur;

    cur->link=next;

}

void insert\_after()

{

    int value;

    cur = (struct node\*)malloc(sizeof(struct node));

    printf("Enter data\n");

    scanf("%d",&(cur->data));

    printf("Enter value\n");

    scanf("%d",&value);

    next=head;

    while(next->data!=value && next!=NULL)

    {

    next=next->link;

    }

    cur->link=next->link;

    next->link=cur;

}

void delete\_at\_begin()

{

    cur=head;

    head=cur->link;

    cur->link=NULL;

    printf("Deleted element is %d\n",cur->data);

    free(cur);

}

void delete\_at\_end()

{

    cur=head;

    while(cur->link!=tail)

    {

    cur=cur->link;

    }

    cur->link=NULL;

    next = tail;

    printf("Deleted element is %d\n",next->data);

    free(next);

    tail=cur;

}

void delete\_at\_pos()

{

    int pos,c=1;

    printf("Enter position of deletion\n");

    scanf("%d",&pos);

    next=head;

    while(c<pos)

    {

    prev=next;

    next=next->link;

    c++;

    }

    prev->link=next->link;

    printf("Deleted element is %d\n",next->data);

    next->link=NULL;

    free(next);

}

void delete\_before()

{

    int value;

    printf("Enter before which node we need to delete\n");

    scanf("%d",&value);

    next=head;

    while(next->link->data != value)

    {

    prev=next;

    next=next->link;

    }

    prev->link = next->link;

    next->link=NULL;

    printf("Deleted element is %d\n",next->data);

    free(next);

}

void delete\_after()

{

    int value;

    printf("Enter after which element we need to delete\n");

    scanf("%d",&value);

    next=head;

    while(next->data!=value)

    {

    prev=next;

    next=next->link;

    }

    prev=next->link;

    next->link=prev->link;

    printf("deleted data is %d\n",prev->data);

    prev->link=NULL;

    free(prev);

}

void display()

{

    if(head==NULL)

    {

    printf("List is empty\n");

    }

    else

    {

    next=head;

    while(next!=NULL)

    {

    printf("%d->",next->data);

    next=next->link;

    }

    }

}

void sort()

{

    struct node\*p1,\*p2;

    p1=head;

    int c=0,i,t;

    while(p1!=NULL)

    {

    c++;

    p1=p1->link;

    }

    for(i=0;i<c;i++)

    {

    p2=head;

    while(p2->link!=NULL)

    {

    if(p2->data>p2->link->data)

    {

    t=p2->data;

    p2->data=p2->link->data;

    p2->link->data=t;

    }

    p2=p2->link;

    }

    }

}

void reverse\_sll()

{

    int a[100],i=0;

    cur=head;

    while(cur!=NULL)

    {

    a[i++]=cur->data;

    cur=cur->link;

    }

    while(i>=0)

    {

    printf("%d->",a[i]);

    i--;

    }

}

int search()

{

     int key,flag=0,c=1;

    printf("Enter value\n");

    scanf("%d",&key);

     cur=head;

     while(cur!=NULL)

     {

             if(cur->data==key)

             {

                     flag=1;

    printf("Element found at %d\n",c);

                     break;

             }

             cur=cur->link;

    c++;

     }

    if(flag==0)

    {

    printf("Element not found\n");

    }

}

int main()

{

    int ch;

    while(1)

    {

    printf("program for single linked list\n");

    printf("1-create\n2-insert at begin\n3-insert at position\n4-insert at end\n5-insert before\n");

    printf("6-insert after\n7-delete at begin\n8-delete at end\n9-delete at pos\n10-delete before\n");

    printf("11-delete after\n12-traversal\n13-display in reverse\n14-search\n15-sort\n");

    printf("enter your choice\n");

    scanf("%d",&ch);

    switch(ch)

    {

    case 1: create();

    break;

    case 2: insert\_at\_begin();

    break;

    case 3: insert\_at\_pos();

    break;

    case 4: insert\_at\_end();

    break;

    case 5: insert\_before();

    break;

    case 6: insert\_after();

    break;

    case 7: delete\_at\_begin();

    break;

    case 8: delete\_at\_end();

    break;

    case 9: delete\_at\_pos();

    break;

    case 10: delete\_before();

      break;

    case 11: delete\_after();

      break;

    case 12: display();

      break;

    case 13: reverse\_sll();

      break;

    case 14: search();

      break;

    case 15: sort();

      break;

    case 16: exit(0);

    }

    }

    return 0;

}

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc linkedList.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**1**

**Enter number of nodes we need to create**

**5**

**current node data**

**10**

**current node data**

**30**

**current node data**

**20**

**current node data**

**40**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->30->20->40->**program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**2**

**Enter data**

**5**

5program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**5->10->30->20->40**->program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**3**

**Enter data**

**35**

**Enter position**

**4**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**5->10->30->35->20->40-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**4**

**enter data**

**50**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**5->10->30->35->20->40->50**->program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**5**

**Enter data**

**15**

**Enter value**

**20**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**5->10->30->35->15->20->40->50**->program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**6**

**Enter data**

**45**

**Enter value**

**40**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**5->10->30->35->15->20->40->45->50-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**7**

**Deleted element is 5**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->30->35->15->20->40->45->50-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**8**

**Deleted element is 50**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**9**

**Enter position of deletion**

**3**

**Deleted element is 35**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->30->15->20->40->45-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**10**

**Enter before which node we need to delete**

**45**

**Deleted element is 40**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->30->15->20->45-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**11**

**Enter after which element we need to delete**

**15**

**deleted data is 20**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->30->15->45**->program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**13**

**-1790676064->45->15->30->10-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**14**

**Enter value**

**30**

**Element found at 2**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->30->15->45-**>program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**15**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**10->15->30->45-**>program for single linked list

**Write a program that uses functions to perform the following operations on Double linked list.**

**i) Creation**

**ii) Insertion**

**iii) Deletion**

**iv) Traversal**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    float data;

    struct node\*prev;

    struct node\*next;

};

struct node\*head=NULL,\*tail=NULL,\*cur,\*t1,\*t2;

void create()

{

    int n;

    printf("Enter the number of nodes\n");

    scanf("%d",&n);

    for(int i=0;i<n;i++)

    {

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter current node data\n");

    scanf("%f",&(cur->data));

    cur->prev=NULL;

    cur->next=NULL;

    if(head==NULL)

    {

    head=tail=cur;

    }

    else

    {

    tail->next=cur;

    cur->prev=tail;

    tail=cur;

    }

    }

}

void insert\_at\_begin()

{

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter current node data\n");

    scanf("%f",&(cur->data));

    cur->prev=NULL;

    cur->next=head;

    head->prev=cur;

    head=cur;

}

void insert\_at\_end()

{

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter cur node data\n");

    scanf("%f",&(cur->data));

    cur->next=NULL;

    cur->prev=tail;

    tail->next=cur;

    tail=cur;

}

void insert\_at\_pos()

{

    int c=1,pos;

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter current node data\n");

    scanf("%f",&(cur->data));

    printf("Enter position\n");

    scanf("%d",&pos);

    t1=head;

    while(c<pos && t1!=NULL)

    {

    t2=t1;

    t1=t1->next;

    c++;

    }

    t2->next=cur;

    cur->prev=t2;

    cur->next=t1;

    t1->prev=cur;

}

void insert\_before()

{

    float value;

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter current node data\n");

    scanf("%f",&(cur->data));

    printf("Enter the data before which we need to insert\n");

    scanf("%f",&value);

    t1=head;

    while(t1!=NULL && t1->data!=value)

    {

    t2=t1;

    t1=t1->next;

    }

    t2->next=cur;

    cur->prev=t2;

    cur->next=t1;

    t1->prev=cur;

}

void insert\_after()

{

    float value;

     cur=(struct node\*)malloc(sizeof(struct node));

     printf("Enter current node data\n");

     scanf("%f",&(cur->data));

     printf("Enter the data after which we need to insert\n");

     scanf("%f",&value);

     t1=head;

     while(t1!=NULL && t1->data!=value)

     {

             t1=t1->next;

     }

     cur->next=t1->next;

    t1->next->prev=cur;

    t1->next=cur;

    cur->prev=t1;

}

void delete\_at\_begin()

{

    cur=head;

    head=head->next;

    head->prev=NULL;

    cur->next=NULL;

    printf("Deleted element is %f\n",cur->data);

    free(cur);

}

void delete\_at\_end()

{

    cur=tail;

     tail=tail->prev;

     tail->next=NULL;

     cur->prev=NULL;

     printf("Deleted element is %f\n",cur->data);

     free(cur);

}

void delete\_at\_pos()

{

    int c=1,pos;

    printf("Enter the position of deletion\n");

    scanf("%d",&pos);

    t1=head;

    while(c<pos && t1!=NULL)

    {

    t2=t1;

    t1=t1->next;

    c++;

    }

    t2->next=t1->next;

    t1->next->prev=t2;

    printf("Deleted data is %f\n",t1->data);

    free(t1);

}

void delete\_before()

{

    float value;

    printf("Enter before which node we need to delete\n");

    scanf("%f",&value);

    t1=head;

    while(t1->next->data!=value && t1->next!=NULL)

    {

    t2=t1;

    t1=t1->next;

    }

    t2->next=t1->next;

    t1->next->prev=t2;

    printf("Deleted element %f\n",t1->data);

    free(t1);

}

void delete\_after()

{

    float value;

    t1=head;

    printf("Enter after which node we need to delete\n");

    scanf("%f",&value);

    while(t1!=NULL && t1->data!=value)

    {

    t2=t1;

    t1=t1->next;

    }

    t2=t1->next;

    t1->next=t2->next;

    t2->next=NULL;

    printf("Deleted element is %f\n",t2->data);

    free(t2);

}

void display()

{

    if(head==NULL)

    {

    printf("DLL is empty\n");

    }

    else

    {

    cur=head;

    while(cur!=NULL)

    {

    printf("%.2f <-> ",cur->data);

    cur=cur->next;

    }

    }

}

void display\_reverse()

{

    if(head==NULL)

    {

    printf("DLL is empty\n");

    }

    else

    {

    cur=tail;

    printf("Elements of dll are\n");

    while(cur!=head)

    {

    printf("%f <->",cur->data);

    cur=cur->prev;

    }

    printf("%f <->",cur->data);

    }

}

void sort()

{

    struct node \*p1,\*p2,\*last=NULL;

    int t,c;

    do

    {

    c=0;

    p1=head;

    while(p1->next!=last)

    {

    if(p1->data > p1->next->data)

    {

    t=p1->data;

    p1->data=p1->next->data;

    p1->next->data=t;

    c=1;

    }

    p1=p1->next;

    }

    last=p1;

    }

    while(c);

}

void search()

{

    float value;

    int  flag=0;

    printf("Enter the value to be searched\n");

    scanf("%f",&value);

    int c=1;

    t1=head;

    while(t1!=NULL)

    {

    if(t1->data == value)

    {

    flag=1;

    break;

    }

    t1=t1->next;

    c++;

    }

    if(flag==1)

    {

    printf("Element present in list at position %d\n",c);

    }

    else

    {

    printf("Element not found\n");

    }

}

int main()

{

     int ch;

     while(1)

     {

             printf("program for single linked list\n");

             printf("1-create\n2-insert at begin\n3-insert at position\n4-insert at end\n5-insert before\n");

             printf("6-insert after\n7-delete at begin\n8-delete at end\n9-delete at pos\n10-delete before\n");

             printf("11-delete after\n12-traversal\n13-display in reverse\n14-search\n15-sort\n");

             printf("enter your choice\n");

             scanf("%d",&ch);

             switch(ch)

             {

                     case 1: create();

                             break;

                     case 2: insert\_at\_begin();

                             break;

                     case 3: insert\_at\_pos();

                             break;

                     case 4: insert\_at\_end();

                             break;

                     case 5: insert\_before();

                             break;

                     case 6: insert\_after();

                             break;

                     case 7: delete\_at\_begin();

                             break;

                     case 8: delete\_at\_end();

                             break;

                     case 9: delete\_at\_pos();

                             break;

                     case 10: delete\_before();

                              break;

                     case 11: delete\_after();

                              break;

                     case 12: display();

                              break;

    case 13: display\_reverse();

                              break;

                     case 14: search();

                              break;

                     case 15: sort();

                              break;

                     case 16: exit(0);

             }

     }

     return 0;

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc dll.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**1**

**Enter the number of nodes**

**4**

**Enter current node data**

**2.2**

**Enter current node data**

**1.5**

**Enter current node data**

**3.3**

**Enter current node data**

**4.1**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 3.30 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**2**

**Enter current node data**

**2.5**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

enter your choice

**12**

**2.50 <-> 2.20 <-> 1.50 <-> 3.30 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**3**

**Enter current node data**

**3.4**

**Enter position**

**4**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.50 <-> 2.20 <-> 1.50 <-> 3.40 <-> 3.30 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**4**

**Enter cur node data**

**1**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.50 <-> 2.20 <-> 1.50 <-> 3.40 <-> 3.30 <-> 4.10 <-> 1.00** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**5**

**Enter current node data**

**3.2**

**Enter the data before which we need to insert**

**4.10**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.50 <-> 2.20 <-> 1.50 <-> 3.40 <-> 3.30 <-> 3.20 <-> 4.10 <-> 1.00** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**6**

**Enter current node data**

**5.4**

**Enter the data after which we need to insert**

**1.50**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**7**

**Deleted element is 2.500000**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 5.40 <-> 3.40 <-> 3.30 <-> 3.20 <-> 4.10 <-> 1.00** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**8**

**Deleted element is 1.000000**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 5.40 <-> 3.40 <-> 3.30 <-> 3.20 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**9**

**Enter the position of deletion**

**3**

**Deleted data is 5.400000**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 3.40 <-> 3.30 <-> 3.20 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**10**

**Enter before which node we need to delete**

**3.20**

**Deleted element 3.300000**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 3.40 <-> 3.20 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**11**

**Enter after which node we need to delete**

**1.50**

**Deleted element is 3.400000**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 3.20 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**13**

**Elements of dll are**

**4.100000 <->3.200000 <->-0.000000 <->1.500000 <->2.20000**0 <->program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 3.20 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**14**

**Enter the value to be searched**

**3.20**

**Element present in list at position 3**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**2.20 <-> 1.50 <-> 3.20 <-> 4.10** <-> program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**15**

program for single linked list

1-create

2-insert at begin

3-insert at position

4-insert at end

5-insert before

6-insert after

7-delete at begin

8-delete at end

9-delete at pos

10-delete before

11-delete after

12-traversal

13-display in reverse

14-search

15-sort

**enter your choice**

**12**

**1.50 <-> 2.00 <-> 3.20 <-> 4.10**

**Write a program that uses functions to perform the following operations on circular linked list: i) Creation ii) Insertion iii) Deletion iv) Traversal**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\*link;

};

struct node\*head=NULL,\*cur, \*temp,\*t1,\*t2;

void create()

{

cur=(struct node\*)malloc(sizeof(struct node));

printf("enter cur data");

scanf("%d",&cur->data);

cur->link=cur;

if(head==NULL)

head=cur;

else

{

temp=head;

while(temp->link!=head)

temp=temp->link;

temp->link=cur;

cur->link=head;

}

}

void insertatbegin()

{

cur=(struct node\*)malloc(sizeof(struct node));

printf("enter cur data");

scanf("%d",&cur->data);

temp=head;

while(temp->link!=head)

temp=temp->link;

temp->link=cur;

cur->link=head;

head=cur; //not required for insert at end

}

void insertatpos()

{

int c=1,pos;

printf("enter position of insertion");

scanf("%d",&pos);

cur=(struct node\*)malloc(sizeof(struct node));

printf("enter cur data");

scanf("%d",&cur->data);

t1=head;

while(c<pos)

{

t2=t1;

t1=t1->link;

c++;

}

t2->link=cur;

cur->link=t1;

}

void deleteatbegin()

{

cur=temp=head;

while(temp->link!=head)

temp=temp->link;

temp->link=cur->link;

head=cur->link;

cur->link=NULL;

printf("deleted element %d",cur->data);

free(cur);

}

void deleteatend()

{

t1=head;

while(t1->link!=head)

{

t2=t1;

t1=t1->link;

}

t2->link=head;

t1->link=NULL;

printf("deleted element %d",t1->data);

free(t1);

}

void deleteatpos()

{

int c=1,pos;

printf("enter position of deletion");

scanf("%d",&pos);

t1=head;

while(c<pos)

{

t2=t1;

t1=t1->link;

c++;

}

t2->link=t1->link;

t1->link=NULL;

printf("deleted element %d",t1->data);

free(t1);

}

void display()

{

if(head==NULL)

printf("CLL is empty");

else

{

temp=head;

while(temp->link!=head)

{

printf("%d->",temp->data);

temp=temp->link;

}

printf("%d",temp->data);

}

}

int main()

{

int ch;

while(1)

{

printf("1-create\n2-insertatbegin\n3-insertatpos\n4-deleteatbegin\n5-deleteatend\n6-delatpos\n7-traversal\n8-exit\n");

printf("enter yr choice");

scanf("%d",&ch);

switch(ch)

{

case 1: create();

break;

case 2: insertatbegin();

break;

case 3: insertatpos();

break;

case 4: deleteatbegin();

break;

case 5: deleteatend();

break;

case 6: deleteatpos();

break;

case 7:display();

break;

case 8: exit(0);

}

}

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ vim cll.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc cll.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

1

enter cur data

10

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

1

enter cur data

20

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

1

enter cur data

30

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

1

enter cur data

40

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

7

10->20->30->40

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

2

enter cur data

5

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

7

5->10->20->30->40

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

3

enter position of insertion

2

enter cur data

15

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

7

5->15->10->20->30->40

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

4

deleted element 5

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

7

15->10->20->30->40

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

5

deleted element 40

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

7

15->10->20->30

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

6

enter position of deletion

3

deleted element 20

1-create

2-insertatbegin

3-insertatpos

4-deleteatbegin

5-deleteatend

6-delatpos

7-traversal

8-exit

enter yr choice

7

15->10->30

##### Write a program that implement stack (its operations) using

##### A. Arrays

##### B. Pointers

##### Implement following operations

##### i) Push

##### ii)Pop

##### iii) Is\_Stack\_Full

##### iv) Is-Stack\_Empty

**A)Arrays**

//stack implementation using arrays

#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

int a[maxsize],top=-1;

void push(int item)

{

    top++;

    a[top]=item;

}

int pop() //return deleted element

{

    return(a[top--]);

}

int isfull()

{

    if(top==maxsize-1)

    {

    return 1;

    }

    else

    {

    return 0;

    }

}

int isempty()

{

    if(top==-1)

    {

    return 1;

    }

    else

    {

    return 0;

    }

}

int peek()

{

    return a[top];

}

void display()

{

    for(int i=top;i>=0;i--)

    {

    printf("%d ",a[i]);

    }

}

int main()

{

    int ch,x;

    while(1)

    {

    printf("\n1-push\n2-pop\n3-display\n4-peek\n5-exit\n");

    printf("enter your choice\n");

    scanf("%d",&ch);

    switch(ch)

    {

    case 1: if(isfull())

    {

    printf("stack overflow\n");

    }

    else

    {

    printf("enter data to push\n");

    scanf("%d",&x);

    push(x);

    }

    break;

    case 2: if(isempty())

    {

    printf("stack underflow\n");

    }

    else

    {

    x=pop();

    printf("deleted element is %d\n",x);

    }

    break;

    case 3: display();

    break;

    case 4: printf("top most element on the stack is %d",peek());

    break;

    case 5: exit(0);

    }

    }

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc stack.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**enter data to push**

**50**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**enter data to push**

**40**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**enter data to push**

**30**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**enter data to push**

**20**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**3**

**20 30 40 50**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**4**

**top most element on the stack is 20**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**3**

**20 30 40 50**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**2**

**deleted element is 20**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**3**

**30 40 50**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**4**

**top most element on the stack is 30**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**5**

**B)Pointers**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node\*link;

};

struct node\*top,\*cur;

void push()

{

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter current node data\n");

    scanf("%d",&(cur->data));

    cur->link=top;

    top=cur;

}

void pop()

{

    cur=top;

    top=cur->link;

    printf("Deleted element is: %d\n",(cur->data));

    cur->link=NULL;

    free(cur);

}

void display()

{

    if(top==NULL)

    {

    printf("Stack underflow\n");

    }

    else

    {

    cur=top;

    printf("Stack->");

    while(cur!=NULL)

    {

    printf("%d->",cur->data);

    cur=cur->link;

    }

    }

}

void peek()

{

    printf("Peek is: %d\n",top->data);

}

int main()

{

     int ch;

     while(1)

     {

             printf("\n1-push\n2-pop\n3-display\n4-peek\n5-exit\n");

             printf("enter your choice\n");

             scanf("%d",&ch);

             switch(ch)

             {

    case 1: push();

    break;

    case 2: pop();

    break;

    case 3: display();

    break;

    case 4: peek();

    break;

    case 5: exit(0);

    }

    }

    return 0;

}

OUTPUT

user@user-Lenovo-G50-80:~$ gcc stacksll.c

user@user-Lenovo-G50-80:~$ ./a.out

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**Enter current node data**

**50**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**Enter current node data**

**30**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**Enter current node data**

**40**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**1**

**Enter current node data**

**20**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**3**

**Stack->20->40->30->50->**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**4**

**Peek is: 20**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**2**

**Deleted element is: 20**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**3**

**Stack->40->30->50->**

1-push

2-pop

3-display

4-peek

5-exit

**enter your choice**

**4**

**Peek is: 40**

1-push

2-pop

3-display

4-peek

5-exit

enter your choice

5

##### Write a C program that implement Queue and its operations using

##### A. Arrays

##### B. Pointers

##### Operations

##### i) Enqueue

##### ii) Dequeue

##### iii)Is\_Queue\_Full

##### iv)Is\_Queue\_Empty

##### A.Arrays

//implementation of queue using arrays

#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

int f=-1,r=-1,queue[maxsize];

int isfull()

{

    if(r==maxsize-1)

    return 1;

    else

    return 0;

}

int isempty()

{

    if(f==-1)

    return 1;

    else

    return 0;

}

void enqueue(int data)

{

    if(f==-1)

    f=0;

    queue[++r]=data;

}

int dequeue()

{

    int x=queue[f];

    if(f==r)

    f=r=-1;

    else

    f++;

    return x;

}

void display()

{

    for(int i=f;i<=r;i++)

    {

    printf("%d\t",queue[i]);

    }

}

int main()

{

    int ch,x;

    while(1)

    {

    printf("\n1-enque\n2-dequeue\n3-display\n4-exit\n");

    printf("Enter your choice\n");

    scanf("%d",&ch);

    switch(ch)

    {

    case 1: if(isfull())

    {

    printf("Queue overflow\n");

    }

    else

    {

    printf("Enter data to insert\n");

    scanf("%d",&x);

    enqueue(x);

    }

    break;

    case 2: if(isempty())

    printf("queue underflow\n");

    else if(f!=r)

    {

    printf("delete element is %d\n",dequeue());

    }

    else if(f==r)

    {

    f=r=-1;

    }

    break;

    case 3: display();

    break;

    case 4: exit(0);

    }

    }

    return 0;

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc queue\_Arrays.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**10**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**20**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**30**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**40**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**50**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Queue overflow**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**3**

**10    20    30    40    50**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**2**

**delete element is 10**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**3**

**20    30    40    50**

1-enque

2-dequeue

3-display

4-exit

**Enter your choice**

**4**

**B.Pointers**

//implementation of queue using sll

#include <stdio.h>

#include <stdlib.h>

struct node

{

    int data;

    struct node\*link;

};

struct node\*f=NULL,\*r=NULL,\*cur;

void enqueue()

{

    cur=(struct node\*)malloc(sizeof(struct node));

    printf("Enter data to insert\n");

    scanf("%d",&cur->data);

    cur->link=NULL;

    if(f==NULL)

    f=r=cur;

    else

    {

    r->link=cur;

    r=cur;

    }

}

void dequeue()

{

    if(f==NULL)

    printf("queue overflow\n");

    else if(f==r)

    {

    printf("deleted data is %d\n",f->data);

    f=r=NULL;

    }

    else

    {

    cur=f;

    f=f->link;

    printf("deleted data %d\n",cur->data);

    free(cur);

    }

}

void display()

{

    if(f==NULL)

    printf("Queue is empty\n");

    else

    {

    cur=f;

    while(cur != NULL)

    {

    printf("%d->",cur->data);

    cur=cur->link;

    }

    }

}

int main()

{

    int ch;

    while(1)

    {

    printf("----------------------------------\n");

    printf("\n1-enqueue\n2-dequeue\n3-display\n4-exit\n");

    printf("Enter your choice\n");

    scanf("%d",&ch);

    switch(ch)

    {

    case 1: enqueue();

    break;

    case 2: dequeue();

    break;

    case 3: display();

    break;

    case 4: exit(0);

    }

    }

}

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc queue\_sll.c

user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out

----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**10**

----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**20**

----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**30**

----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**1**

**Enter data to insert**

**40**

----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**3**

**10->20->30->40**->----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**2**

**deleted data 10**

----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

**Enter your choice**

**3**

**20->30->40->**----------------------------------

1-enqueue

2-dequeue

3-display

4-exit

Enter your choice

4

**Write a program to implement the graph traversal method**

**i)DFS**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int source,V,E,time,visited[20],G[20][20];  void DFS(int i)  {  int j;  visited[i]=1;  printf(" %d->",i+1);  for(j=0;j<V;j++)  {  if(G[i][j]==1&&visited[j]==0)  DFS(j);  }  }  int main()  {  int i,j,v1,v2;  printf("\t\t\tGraphs\n");  printf("Enter the no of edges:");  scanf("%d",&E);  printf("Enter the no of vertices:");  scanf("%d",&V);  for(i=0;i<V;i++)  {  for(j=0;j<V;j++)  G[i][j]=0;  }  /\* creating edges :P \*/  for(i=0;i<E;i++)  {  printf("Enter the edges (format: V1 V2) : ");  scanf("%d%d",&v1,&v2);  G[v1-1][v2-1]=1;  }  for(i=0;i<V;i++)  {  for(j=0;j<V;j++)  printf(" %d ",G[i][j]);  printf("\n");  }  printf("Enter the source: ");  scanf("%d",&source);  DFS(source-1);  return 0;  } |
|  |
|  |  |

**OUTPUT**

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc dfs.c  
user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out  
Graphs  
Enter the number of edges:  
8  
Enter number 0f vertices:  
4  
Enter edge (V1 V2) format:  1 3  
Enter edge (V1 V2) format:  3 1  
Enter edge (V1 V2) format:  1 2  
Enter edge (V1 V2) format:  2 1  
Enter edge (V1 V2) format:  2 4  
Enter edge (V1 V2) format:  4 2  
Enter edge (V1 V2) format:  3 4  
Enter edge (V1 V2) format:  4 3  
0 1 1 0  
1 0 0 1  
1 0 0 1  
0 1 1 0  
Enter the source: 1  
1 -> 2 -> 4 -> 3 ->

**ii)BFS**

#include<stdio.h>

int G[20][20],q[20],visited[20],n,front = 1, rear = 0 ;

void bfs(int v)

{

int i;

visited[v] = 1;

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

if(G[v][i] && !visited[i])

q[++rear]=i;

if(front <= rear)

bfs(q[front++]);

}

int main()

{

int v,i,j;

printf("\n Enter the number of vertices:");

scanf("%d",&n);

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

{

q[i]=0;

visited[i]=0;

}

printf("\n Enter graph data in matrix form:\n");

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

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

scanf("%d",&G[i][j]);

printf("\n Enter the starting vertex:");

scanf("%d",&v);

bfs(v);

printf("\n The nodes which are reachable are:\n");

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

if(visited[i])

printf("%d\t",i);

else

printf("\n %d is not reachable",i);

return 0;

}

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc bfs.c  
user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out  
  
 Enter the number of verticees:4  
  
Enter graph data in matrix form:  
0 1 1 0  
1 0 0 1  
1 0 0 1  
0 1 1 0  
  
Enter the starting vertex:1  
  
 The node which are reachable are:  
1 2 3 4

**Write a program to implement the tree traversal methods**

|  |
| --- |
|  |
|  | #include<string.h> |
|  |  |
|  | #include<stdio.h>  struct BST |
|  | { |
|  | int data; |
|  | struct BST \*left,\*right; |
|  | }; |
|  | struct BST\* root=NULL,\*temp,\*cur; |
|  | void create() |
|  | { |
|  | temp = root; |
|  | cur = (struct BST\*)malloc(sizeof(struct BST)); |
|  | printf("Enter data\n"); |
|  | scanf("%d",&(cur->data)); |
|  | cur->left = NULL; |
|  | cur->right = NULL; |
|  | if(temp==NULL) |
|  | { |
|  | root=cur; |
|  | } |
|  | else |
|  | { |
|  | while(temp!=NULL) |
|  | { |
|  | if((cur->data)<(temp->data)) |
|  | { |
|  | if(temp->left==NULL) |
|  | { |
|  | temp->left=cur; |
|  | return; |
|  | } |
|  | else |
|  | { |
|  | temp=temp->left; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | if(temp->right==NULL) |
|  | { |
|  | temp->right=cur; |
|  | return; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | void postorder(struct BST \*temp) |
|  | { |
|  | if(temp!=NULL) |
|  | { |
|  | postorder(temp->left); |
|  | postorder(temp->right); |
|  | printf("%d ",temp->data); |
|  | } |
|  | } |
|  | void preorder(struct BST\* temp) |
|  | { |
|  | if(temp!=NULL) |
|  | { |
|  | printf("%d ",temp->data); |
|  | preorder(temp->left); |
|  | preorder(temp->right); |
|  | } |
|  | } |
|  | void inorder(struct BST\* temp) |
|  | { |
|  | if(temp!=NULL) |
|  | { |
|  | inorder(temp->left); |
|  | printf("%d ",temp->data); |
|  | inorder(temp->right); |
|  | } |
|  | } |
|  | int main() |
|  | { |
|  | int ch; |
|  | while(1) |
|  | { |
|  | printf("\n1-Create\n2-Preorder\n3-Inorder\n4-PostOrder\n5-Exit\n"); |
|  |  |
|  | printf("Enter your choice\n"); |
|  | scanf("%d",&ch); |
|  | switch(ch) |
|  | { |
|  | case 1:create(); |
|  | break; |
|  | case 2:printf("Preorder Traversal\n"); |
|  | preorder(root); |
|  | break; |
|  | case 3:printf("Inorder Traversal\n"); |
|  | inorder(root); |
|  | break; |
|  | case 4:printf("Postorder Traversal\n"); |
|  | postorder(root); |
|  | break; |
|  | case 5:exit(0); |
|  | break; |
|  | default:printf("Invalid Choice\n"); |
|  | } |
|  | } |
|  | return 0; |
|  | } |

OUTPUT

user@user-Lenovo-G50-80:~/Desktop/dslab$ gcc bin\_searchTree.c  
user@user-Lenovo-G50-80:~/Desktop/dslab$ ./a.out  
  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
1  
Enter data  
25  
  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
1  
Enter data  
22  
  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
1  
Enter data  
34  
  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
2  
Preorder Traversal  
25 22 34  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
3  
Inorder Traversal  
22 25 34  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
4  
Postorder Traversal  
22 34 25  
1-Create  
2-Preorder  
3-Inorder  
4-PostOrder  
5-Exit  
Enter your choice  
5

|  |
| --- |
|  |