|  |  |
| --- | --- |
| **14** | **WRITE A PROGRAM TO IMPLEMENT SINGLY LINKLIST.** |
| **15** | **WRITE A PROGRAM TO IMPLEMENT DOUBLY LINKLIST.** |
| **16** | **WRITE A PROGRAM TO IMPLEMENT BUBBOLE SORT.** |
| **17** | **WRITE A PROGRAM TO IMPLEMENT HEAP SORT.** |
| **18** | **WRITE A PROGRAM TO IMPLEMENT QUICK SORT.** |
| **19** | **WRITE A PROGRAM TO IMPLEMENT INSERTION SORT.** |
| **20** | **WRITE A PROGRAM TO IMPLEMENT SELECTION SORT.** |
| **21** | **WRITE A PROGRAM TO IMPLEMENT LINEAR SEARCHING** |
| **22**  **23** | **WRITE A PROGRAM TO IMPLEMENT TO WAY MERGE SORT**  **WRITE A PROGRAM TO IMPLEMENT BINARY SEARCHING.** |

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : Write a algorithm to impalement singly link list.

**NAME** :RAVAL JATIN

**CLASS**  : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**ADD TO BEGING()**

**Step 1 :** Start

**Step 2 :** [ Allocate free space ]

Node = caret new node

**Step 3 :** [ Read vale of information part of a new node ]

Node[ info ] = data

**Step 4 :** [ Link address part of the currently created node with the address of start ]

Node [ next ] = start

**Step 5 :** [ Now assign address of newly created node to the start ]

Start = node

**Step 6 :** Exit

**ADD TO LAST()**

**Step 1 :** Start

**Step 2 :** [ Here start contain the first node address initially list empty ]

Start = NULL

**Step 3 :** [ Allocation space to newly create node ]

Node = create a new node

**Step 4 :** [ Assign value to information part of node ]

Node [ info ] = data

**Step 5 :** [Assign NULL to the address part of new created node ]

Node [ next ] = NULL

**Step 6 :** [ Assign address of the node to it’s previous variable ]

IF start = NULL than start = node

Else [ Move until last node is achieved ]

Temp = start

Repeated while temp [ next ] != NULL

temp = temp [ next ]

temp [ next ] = node

**Step 7 :** Exit

**ADD TO POS()**

**Step 1:** start

**Step 2:** [create and initialize of node]

Node =create a new node

temp = start

**Step 3:** [read node number position that we want to insert]

Input node\_no

**Step 4:** [perform insertion operation]

Repeat through step 5 until does not reach to node\_no

Check if temp = NULL then

Printf ”you inputted wrong node number”

Return

temp = temp[next]

**Step 5:** [insert node ]

New\_Nnode = create a new node

New\_Nnode[info] = data

New\_Nnode[next]=temp[next]

temp[next] = New\_Node

**step 6:** Exit

**ADD TO INFO()**

**Step 1:** start

**Step 2:** [create and initialize of node]

Node =create a new node

temp = start

**Step 3:** [read data information at which location we want to insert]

Input ins\_data

**Step 4:** [perform insertion operation]

Repeat until temp[info]!= ins\_data

Check if temp = NULL then

Printf ”you inputted wrong data value”

Return

temp = temp[next]

**Step 5:** [insert node ]

New\_Nnode = create a new node

New\_Nnode[info] = data

New\_Nnode[next]=temp[next]

temp[next] = New\_Node

**Step 6:** Exit

**DELETE()**

**Step 1:** start

**Step 2:** [assign start address of link list to another node]

node = start

**Step 3:** [perform deletion operation]

if Node = NULL

output”UNDERFLOW”

exit

else [delete first node]

start = node[next]

delete node

**Step 4:** Exit

**DIS()**

**Step 1 :** Start

**Step 2 :** [ Initialized pointer variable node ]

Node = start

**Step 3 :** [ Perform the traverse operation ]

Repeat while node [ next ] != NULL

Display node [ info ]

**Step 4 :** [ Move pointer to next node ]

Node = node [ next ]

**Step 5 :** Exit

**SORT() link list**

**step 1:** start

**step 2:** create node q and t

assign start address of the link to q

q=start

**step 3:** repeat step 4 until q[next] not Null

q=q[next]

**step 4:** assign a[next] to t

t=q[next]

repeat these until t is not Null

t=t[next]

**step 5:** check the data of q and t

if q[data] > t[data] then

swap q[data] with t[data]

**step 6:** Exit

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : Write a program to impalement singly link list.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

struct slink

{

int data;

struct slink \*next;

};

typedef struct slink sl;

void addb(int);

void addl(int);

void addpos(int,int);

void addinfo(int,int);

void del(int);

void dis();

void sort();

void exit();

sl \*start=NULL;

void main()

{

int x,ch,ps;

clrscr();

do

{

printf("\n1. insert at the begining ");

printf("\n2. insert at the last ");

printf("\n3. insert at the specific location ");

printf("\n4. insert based on information ");

printf("\n5. delete ");

printf("\n6. display the link list ");

printf("\n7. sorting of linked list");

printf("\n8. exit");

printf("\nenter your choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1 : printf("enter the value to insert :");

scanf("%d",&x);

addb(x);

dis();

getch();

break;

case 2 : printf("enter the value to insert at last :");

scanf("%d",&x);

addl(x);

dis();

getch();

break;

case 3 : printf("enter the value to insert :");

scanf("%d",&x);

printf("enter the position :");

scanf("%d",&ps);

addpos(x,ps);

dis();

getch();

break;

case 4 : printf("enter the value to insert :");

scanf("%d",&x);

printf("enter the position :");

scanf("%d",&ps);

addinfo(x,ps);

dis();

getch();

break;

case 5 : printf("enter the value to delete :");

scanf("%d",&x);

del(x);

dis();

break;

case 6 : if(start==NULL)

printf("linked list is empty ");

else

dis();

break;

case 7 : if(start==NULL)

printf("linked list is empty ");

else

{

sort();

dis();

getch();

}

break;

case 8 : exit(0);

break;

default : printf("invalid choice ");

break;

}

}while(ch!=8);

getch();

}

void addb(int a)

{

sl \*q;

if(start==NULL)

{

start=(sl\*)malloc(sizeof(sl));

start->data=a;

start->next=NULL;

}

else

{

q=(sl\*)malloc(sizeof(sl));

q->data=a;

q->next=start;

start=q;

}

}

void addl(int x)

{

if(start==NULL)

{

start=(sl\*)malloc(sizeof(sl));

start->data=x;

start->next=NULL;

}

else

{

sl \*q,\*t;

q=start;

while(q->next!=NULL)

{

q=q->next;

}

t=(sl\*)malloc(sizeof(sl));

t->data=x;

t->next=NULL;

q->next=t;

}

}

void addpos(int x,int pos)

{

sl \*q,\*t,pre;

int i;

q=start;

if(pos==1)

{

addb(x);

return;

}

for(i=1;i<pos-1;i++)

{

if(q==NULL)

{

printf("\n link list is empty ");

return;

}

q=q->next;

}

t=(sl\*)malloc(sizeof(sl));

t->data=x;

t->next=q->next;

q->next=t;

}

void addinfo(int x,int ps)

{

sl \*q,\*t;

q=start;

while(q->data!=ps)

{

if(q==NULL)

{

printf("value not found ");

return;

}

q=q->next;

}

t=(sl\*)malloc(sizeof(sl));

t->data=x;

t->next=q->next;

q->next=t;

}

void del(int ps)

{

sl \*q,\*t,\*pre;

int i;

q=start;

if(q==NULL)

{

printf("\n link list is empty ");

}

else

{

if(ps==1)

{

start=q->next;

return;

}

for(i=0;i<ps-1;i++)

{

pre=q;

q=q->next;

}

pre->next=q->next;

}

}

void dis()

{

sl \*q;

for(q=start;q!=NULL;q=q->next)

{

printf("%d\t",q->data);

}

getch();

}

void sort()

{

sl \*q,\*t;

int s;

for(q=start;q->next!=NULL;q=q->next)

{

for(t=q->next;t!=NULL;t=t->next)

{

if(q->data > t->data)

{

s=q->data;

q->data=t->data;

t->data=s;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 1

enter the value to insert :10

10

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 2

enter the value to insert at last :32

10 32

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 3

enter the value to insert :30

enter the position :3

10 32 30

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : Write a algorithm to implement doubly linked list.

**NAME** :RAVAL JATIN

**CLASS**  : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**INSERT AT THE BEGINING**

**Step 1 :** Start.

**Step 2 :** [Allocate free space]

Node = Create a new node

**Step 3 :** [Read the value of information part of a new node]

Node[info] = data

**Step 4 :** [Linked address part of currently created node with the address of start]

Node[next] = start

Start[previous] = node

Node[previous] = NULL

**Step 5 :** [Now assign address of newly created node with the address of start]

Start = node

**Step 6 :** End.

**INSERT AT THE LAST**

**Step 1 :** Start.

**Step 2 :** [ Hear start contains the first node address ; initialized list is empty]

Start = NULL

**Step 3 :** [Allocate space to newly created node]

Node = Create a new node

**Step 4 :** [Assign value to information part of node]

Node[info] = data

**Step 5 : [** Assign null to address part of signalling the end of the list]

Node[previous] = NULL

Node[next] = NULL

**Step 6 : [** Assign address of the node to its previous veriable]

If start == NULL then

Start = node

Else [move until last node is achived]

Temp =start

Repeat while temp[next]!= NULL

Temp = temp[next]

Temp[next] = node

Node[previous] = temp

**Sept 7 :** End.

**ADD AT THE SPECIFIC LOCATION**

**Step 1 :** Start.

**Step 2 :** [Create and initialization of node]

Node = create a new node

Temp = start

**Step 3 :** [ Read node number position that we won’t to insert]

Input node\_no

**Step 4 : [** Perform insertion operation]

Repeat through step 5 until does not reach node\_no

Check if temp ==NULL then

“you inputted wrong number”

Return

Temp = temp[next]

**Step 5 :** [ insert node]

New\_node = create a new node

New\_node[info] = data

New\_node[next] = temp[next]

Temp[next][previous] = new\_node

Temp[next] = new\_node

New[previous] = temp

**Step 6 :** End.

**ADD BASED ON INFORMATION**

**Step 1 :** Start.

**Step 2 :** [Create and initialization of node]

Node = create a new node

Temp = start

**Step 3 :** [ Read data information at which location we won’t to insert]

Input insert\_data

**Step 4 :** [ perform insertion operation ]

Repeat until temp[info] != insert\_data

Check if temp == NULL then

Display “ you inputted wrong information”

Return

Temp = temp[next]

**Step 5 :** [Insert node]

New\_node = create a new node

New\_node [info] = data

New\_node[node] = temp[next]

Temp[next][previous] = new\_node

Temp[next] = new\_node

New\_node[previous] = temp

**Step 6 :** End.

**TRAVERS THE NODE**

**Step 1 :** Start.

**Step 2 :** [initialization pointer variable node]

Temp = start

**Step 3 : [**perform the traversing operation ]

Repeat while temp[next] != NULL

Display temp[info]

**Step 4 :** [move pointer to next node]

Temp = temp[next]

**Step 5 :** End.

**DELETE THE NODE**

**Step 1 :** Start.

**Step 2 :** [assign start address of link list]

Node = start

**Step 3 : [**perform deletion operation ]

If node == NULL then

Output “ Underflow”

Exit

Else [delete first node]

Start = node[next]

Start [previous] = NULL

**Step 4 :** Exit.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : Write a program to implement doubly linked list.

**NAME**  : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

struct dlink

{

int data;

struct dlink \*next;

struct dlink \*pre;

};

typedef struct dlink dl;

void addb(int);

void addl(int);

void addpos(int,int);

void addinfo(int,int);

void del(int);

void dis();

void sort();

void exit();

dl \*start=NULL;

void main()

{

int x,ch,ps;

clrscr();

do

{

printf("1. insert at the begining ");

printf("\n2. insert at the last ");

printf("\n3. insert at the specific location ");

printf("\n4. insert based on information ");

printf("\n5. delete ");

printf("\n6. display the link list ");

printf("\n7. sorting of linked list");

printf("\n8. exit");

printf("\nenter your choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1 : printf("enter the value to insert :");

scanf("%d",&x);

addb(x);

dis();

getch();

break;

case 2 : printf("enter the value to insert at last :");

scanf("%d",&x);

addl(x);

dis();

getch();

break;

case 3 : printf("enter the value to insert :");

scanf("%d",&x);

printf("enter the position :");

scanf("%d",&ps);

addpos(x,ps);

dis();

getch();

break;

case 4 : printf("enter the value to insert :");

scanf("%d",&x);

printf("enter the position value :");

scanf("%d",&ps);

addinfo(x,ps);

dis();

getch();

break;

case 5 : printf("enter the position number to delete :");

scanf("%d",&x);

del(x);

dis();

break;

case 6 : if(start==NULL)

{

printf("linked list is empty ");

}

else

dis();

break;

case 7 : if(start==NULL)

{

printf("linked list is empty ");

}

else

{

sort();

dis();

getch();

}

break;

case 8 : exit(0);

break;

default : printf("invalid choice ");

break;

}

}while(ch!=8);

getch();

}

void addb(int a)

{

dl \*q;

if(start==NULL)

{

start=(dl\*)malloc(sizeof(dl));

start->data=a;

start->next=NULL;

start->pre=NULL;

}

else

{

q=(dl\*)malloc(sizeof(dl));

q->data=a;

q->next=start;

q->pre=q;

start=q;

}

}

void addl(int x)

{

if(start==NULL)

{

start=(dl\*)malloc(sizeof(dl));

start->data=x;

start->next=NULL;

start->pre=NULL;

}

else

{

dl \*q,\*t;

q=start;

while(q->next!=NULL)

{

q=q->next;

}

t=(dl\*)malloc(sizeof(dl));

t->data=x;

t->next=NULL;

t->pre=q;

q->next=t;

}

}

void addpos(int x,int pos)

{

dl \*q,\*t,pre;

int i;

q=start;

if(pos==1)

{

addb(x);

return;

}

for(i=1;i<pos-1;i++)

{

if(q==NULL)

{

printf("\n link list is empty ");

return;

}

q=q->next;

}

t=(dl\*)malloc(sizeof(dl));

t->data=x;

t->next=q->next;

t->pre=q;

q->next=t;

}

void addinfo(int x,int ps)

{

dl \*q,\*t;

q=start;

while(q->data!=ps)

{

if(q==NULL)

{

printf("value not found ");

return;

}

q=q->next;

}

t=(dl\*)malloc(sizeof(dl));

t->data=x;

t->next=q->next;

t->pre=q;

q->next=t;

}

void del(int ps)

{

dl \*q,\*t,\*pre;

int i;

q=start;

if(q==NULL)

{

printf("\n link list is empty ");

}

else

{

if(ps==1)

{

start=q->next;

return;

}

for(i=0;i<ps-1;i++)

{

pre=q;

q=q->next;

}

pre->next=q->next;

}

}

void dis()

{

dl \*q;

for(q=start;q!=NULL;q=q->next)

{

printf("%d\t",q->data);

}

getch();

}

void sort()

{

dl \*q,\*t;

int s;

for(q=start;q->next!=NULL;q=q->next)

{

for(t=q->next;t!=NULL;t=t->next)

{

if(q->data > t->data)

{

s=q->data;

q->data=t->data;

t->data=s;

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 1

enter the value to insert :6

6

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 2

enter the value to insert at last :3

6 3

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 3

enter the value to insert :12

enter the position :2

6 12 3

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 4

enter the value to insert :10

enter the position value :6

6 10 12 3

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 5

enter the position number to delete :3

6 10 3

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 6

6 10 3

1. insert at the begining

2. insert at the last

3. insert at the specific location

4. insert based on information

5. delete

6. display the link list

7. sorting of linked list

8. exit

enter your choice : 7

3 6 10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**AGORITHM** : Write a algorithm to implement bubble sort.

**NAME** : RAVAL JATIN

**CLASS**  : SY BC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Step 1 :** Start

**Step 2 :** Starting with the first element(index = 0), compare the current element

with the next element of the array.

**Step 3 :** If the current element is greater than the next element of the array,

swap them.

**Step 4 :** If the current element is less than the next element, move to the

next element. Repeat Step 1.

**Step 5 :** End

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : Write a program to implement bubble sort.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

void main()

{

int arr[5]={25,17,31,13,2};

int i,j,temp;

clrscr();

printf("BUBBLE SORT\n\n");

printf("Array before sorting :\n");

for(i=0;i<=4;i++)

printf("%d\t",arr[i]);

for(i=0;i<=3;i++)

{

for(j=0;j<=3-i;j++)

{

if(arr[j]>arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("\nArray after sorting :\n");

for(i=0;i<=4;i++)

printf("%d\t",arr[i]);

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BUBBLE SORT

Array before sorting :

25 17 31 13 2

Array after sorting :

2 13 17 25 31

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : Write a program to impalement quick sort.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Step 1** : start

**Step 2** : Choose the highest index value has pivot

**Step 3** : Take two variables to point left and right of the list excluding pivot

**Step 4** : left points to the low index

**Step 5** : right points to the high

**Step 6** : while value at left is less than pivot move right

**Step 7** : while value at right is greater than pivot move left

**Step 8** : if both step 5 and step 6 does not match swap left and right

**Step 9** : if left = right, the point where they met is new pivot

**Step10** : end

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***PROGRAM** : Write a program to impalement quick sort.

**NAME** :Raval jatin

**CLASS**  : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#include<stdio.h>

#include<conio.h>

void quicksort(int\*,int,int);

int split(int\*,int,int);

void main()

{

int array[10]={11,2,9,13,57,25,17,1,90,3};

int i;

clrscr();

printf("QUICK SORT \n\n");

printf("Array before sorting \n");

for(i=0;i<=9;i++)

{

printf("%d\t",array[i]);

}

quicksort(array,0,9);

printf("Array after sorting \n");

for(i=0;i<=9;i++)

{

printf("%d\t",array[i]);

}

getch();

}

void quicksort(int a[],int lower,int upper)

{

int i;

if(upper>lower)

{

i=split(a,lower,upper);

quicksort(a,lower,i-1);

quicksort(a,i+1,upper);

}

}

int split(int a[],int lower,int upper)

{

int p,q,t,i;

p=lower+1;

q=upper;

i=a[lower];

while(q>=p)

{

while(a[p]<i)

p++;

while(a[q]>i)

q--;

if(q>p)

{

t=a[p];

a[p]=a[q];

a[q]=t;

}

}

t=a[lower];

a[lower]=a[q];

a[q]=t;

return q;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUICK SORT

Array before sorting

11 2 9 13 57 25 17 1 90 3

Array after sorting

1 2 3 9 11 13 17 25 57 90

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : Write a algorithm to insertion sort.

**NAME** : RAVAL JATIN

**CLASS**  : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Step 1** : start

**Step 2** : If it is the first element, it is already sorted. return 1;

**Step 3** : Pick next element

**Step 4** : Compare with all elements in the sorted sub-list

**Step 5** : Shift all the elements in the sorted sub-list that is greater than the

value to be sorted

**Step 6** : Insert the value

**Step 7** : Repeat until list is sorted.

**Step 8** : end

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : Write a program to insertion sort.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

void main()

{

int arr[5]={25,17,31,13,2};

int i,j,k,temp;

clrscr();

printf("INSERTION SORT\n\nArray before sorting\n");

for (i=0;i<=4;i++)

{

printf("%d\t",arr[i]);

}

for(i=1;i<=4;i++)

{

for(j=0;j<i;j++)

{

if(arr[j]>arr[i])

{

temp=arr[j];

arr[j]=arr[i];

for (k=i;k>j;k--)

arr[k]=arr[k-1];

arr[k+1]=temp;

}

}

}

printf("\nArray after sorting:\n");

for (i=0;i<=4;i++)

printf("%d\t",arr[i]);

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INSERTION SORT

Array before sorting

25 17 31 13 2

Array after sorting:

2 13 17 25 31

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : Write a algorithm to implement selection sort.

**NAME**  : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**Step 1 :** start

**Step 2 :** Set MIN to location 0

**Step 3 :** Search the minimum element in the list

**Step 4 :** Swap with value at location MIN

**Step 5 :** Increment MIN to point to next element

**Step 6 :** Repeat until list is sorted

**Step 7 :** end

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : Write a program to implement selection sort.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

void main()

{

int arr[5]={25,17,31,13,2};

int i,j,temp;

printf("SELECTION SORT\n\n");

printf("Array before sorting :\n");

for(i=0;i<=4;i++)

printf("%d\t",arr[i]);

for(i=0;i<=3;i++)

{

for(j=i+1;j<=4;j++)

{

if(arr[i]>arr[j])

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

printf("\nArray after sorting :\n");

for(i=0;i<=4;i++)

printf("%d\t",arr[i]);

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SELECTION SORT

Array before sorting :

25 17 31 13 2

Array after sorting :

2 13 17 25 31

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : write a algorithm to implement linear search on a sorted array.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Step 1 :**Start

**Step 2 :** Read the search element from the user

**Step 3 :** Compare, the search element with the first element in the list.

**Step 4 :** If both are matching, then flag=1.

**Step 5 :** If both are not matching, then compare search element with

the next element in the list.

**Step 6 :** Repeat steps 3 and 4 until the search element is compared with

the last element in the list.

**Step 7 :** If the last element in the list then flag = 0,display "Element not found!!!"

and terminate the function.

**Step 8 :**End

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : write a program to implement linear search on a sorted array.

**NAME**  : RAVAL JATIN

**CLASS**  : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

void main()

{

int a[]={10,40,60,20,30,100,50,70,90,80},i,j,f=0,x;

printf("enter the searching number : ");

scanf("%d",&x);

j=x+1;

for(i=0;a[i]<x;i++)

{

if(a[i]==x)

{

f=1;

break;

}

}

if(f==1)

printf("value is found ");

else

printf("value is not found ");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

enter the searching number : 30

value is not found

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : Write a algorithm to implement two way merge sort.

**NAME** : RAVAL JATIN

**CLASS**  : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Step 1 :** start

**Step 2 :** if it is only one element in the list it is already sorted, return.

**Step 3 :** divide the list recursively into two halves until it can no more be divided.

**Step 4 :** merge the smaller lists into new list in sorted order.

**Step 5 :** end

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : Write a program to implement two way merge sort.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

void main()

{

int a[5]={11,2,9,13,57}, b[5]={25,17,1,90,3}, c[10],i,j,k,temp;

clrscr();

printf("MARGE SORT \n\nFirst array\n");

for (i=0;i<=4;i++)

printf("%d\t",a[i]);

printf("\n\nSecond array \n");

for(i=0;i<=4;i++)

printf("%d\t",b[i]);

for(i=0;i<=3;i++)

{

for (j=i+1;j<=4;j++)

{

if(a[i]>a[j])

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

if(b[i]>b[j])

{

temp=b[i];

b[i]=b[j];

b[j]=temp;

}

}

}

for (i=j=k=0;i<=9;)

{

if(a[j]<=b[k])

c[i++]=a[j++];

else

c[i++]=b[k++];

if(j==5 || k==5)

break;

}

for(;j<=4;)

c[i++]=a[j++];

for(;k<=4;)

c[i++]=b[k++];

printf("\n\nArray after sorting\n");

for(i=0;i<=9;i++)

printf("%d\t",c[i]);

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MARGE SORT

First array

11 2 9 13 57

Second array

25 17 1 90 3

Array after sorting

1 2 3 9 11 13 17 25 57 90

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ALGORITHM** : write a algorithm to implement binary search.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Step 1 :**Start

**Step 2 :** Read the search element from the user

**Step 3 :** Find the middle element in the sorted list

**Step 4 :** Compare, the search element with the middle element in the sorted list.

**Step 5 :** If both are matching, then display "Given element found!!!" and

terminate the function

**Step 6 :** If both are not matching, then check whether the search element

is smaller or larger than middle element.

**Step 7 :** If the search element is smaller than middle element,then repeat

steps 2, 3, 4 and 5 for the left sublist of the middle element.

**Step 8 :** If the search element is large than middle element, then repeat

steps 2, 3, 4 and 5 for the right sublist of the middle element.

**Step 9 :** Repeat the same process until we find the search element in

the list or until sublist contains only one element.

**Step 10 :** If that element also doesn't match with the search element,

then display "Element not found in the list!!!" and terminate

the function.

**Step 11 :**End

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PROGRAM** : write a program to implement binary search.

**NAME** : RAVAL JATIN

**CLASS** : SY BCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<conio.h>

void main()

{

int i,num,mid,lower=0,upper=9,flag=1;

int a[10]={1,2,3,4,11,13,17,25,57,90};

clrscr();

printf("Enter the search value : ");

scanf("%d",&num);

for(mid=(lower+upper)/2;lower<=9;mid=(lower+upper)/2)

{

if(a[mid]==num)

{

printf("the number is at position %d in the array",mid);

flag=0;

break;

}

if(a[mid]>num)

{

upper = mid - 1;

}

else

{

lower = mid + 1;

}

}

if(flag==1)

{

printf("Element is not present in the array");

}

getch();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter the search value : 11

the number is at position 4 in the array

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* /