ALGORITHM inputEdge (top,source,dest)

Pre-top address of vertex

Source

Destination

Post-

loop(till i less than count)

if(compare source with name[i])

set x to i

break

end loop

if(I is equal to count)

print Invalid edge

return

end if

loop(till j less than count)

if(compare source with name[j])

y=j

break

end if

end loop

if(j is equal count)

printf("\nInvalid edge")

return;

End if

addedge(top,x,y);

return;

end inputEdge

ALGORITHM input(top,source)

Pre-top of the vertex

Source

Post-vertex is added

name[count]is set to source

top is set to addvertex(top,count)

count++

end input

ALGORITHM displayinput(top)

Pre-top of the vertex

Post-the graph is displayed on console

Set to top;

loop(a is not NULL)

Print name [datav of a]

Set b is name[nnext of a]

loop(b is not NULL)

print name[data of v]

set b b->next

end loop

set a to a->vnext;

print new line

end loop

end displayinput

ALGORITHM addvertex(top, d)

Pre-top of the vertex

Post-vertex is added

Set a to top;

Allocate(new1)

Set new1->vnext to NULL

Set new1->nnext to NULL

Set new1->datav to d

if(top is NULL)

set top to new1

return top;

else

loop(a->vnext is not NULL)

a=a->vnext;

end loop

set a->vnext to new1

end if

return top;

end addvertex

ALGORITHM addedge(top, x,y)

Pre-top of the vertex

Post-edge is added

Allocate(new2)

Set new2 next to NULL

Set new2 data to y

loop(a)

if(a->datav==x){

break

end if

set a to a->vnext

end loop

if(a is NULL)

print Invalid node

return

else

if(a->nnext is NULL)

set a->nnext=new2

else

set a1 is a->nnext

loop(a1->next is NULL)

a1=a1->next

end loop

set a1->next to new2

end addedges

ALGORITHM vertexsearch(top,stop,source,dest)

Pre-top of the vertex

Stop is top of stack

Post-return the searched vertex

Set a to top;

Loop (till d is des)

display(stop)

print new line

if(stopis equal to NULL)

exit(0)

stop=pops(stop,n);

set d to stop->data;

end loop

loop(a is not equal to NULL)

if(a->datav is equal to d)

path(top,stop, a->nnext, des);

end if

a=a->vnext;

end vertexSearch

ALGORITHM path(top,stop,sorce,dest)

Pre-top of the vertex

Stop is top of the stack

Set b to a

Set y to b->data;

Set a to a->next;

loop(a is not NULL)

if(a->data is 0)

stop=push(stop,-100);

else

stop=push(stop, - a->data);

set a is a->next;

end loop

stop=push(stop,b->data);

stop= check(stop);

if(stop is not NULL)

h=stop;

set y to h->data;

vertexSearch( top,stop,y,des);

end path

ALGORITHM pops(stop,n)

Pre-stop is top of stack

loop(stop)

if(stop->data is less than 0)

break;

stop=stop->snext;

end if

if(stop is NULL)

exit(0);

if(stop->data is -100)

set stop->data+=100;

else

set stop->data to -stop->data;

stop=check(stop);

set n to stop->data;

return stop;

end pops

ALGORITHM display(stop)

Pre-stop is top of stack

Set a to stop

Loop(a is not NULL)

Set Arr[i] to a->data

Set a to a->snext

I++

End loop

Loop()

If( arr[j] is less than equal to 0)

Print name[arr[j]]

End display

ALGORITHM push(top,dat)

Pre-top of the vertex

Data to be pushed

Post-data is pushed

Allocate(a);

Set data of a to dat

If( stop is NULL )

Set Stop to a

Return stop

End if

Set a->snext to stop  
set stop to a

Return stop

End push