**Stacks Introduction**

How to free allocated memory in linked list

using while loop(eg:Pooja,Gunashree)

To see if the memory is freed up or not use valgrind

STACK:Last in First out(LIFO):

From top if i am able to see bottom then it is empty

The stack will grow from bottom to top

when Bottom=Top the container is empty

when top=limit the container is full

The size/Limit/max of stack is fixed.

In push operation the top increases

In pop operation the top shrinks

How to avoid multiple defintions of variables

/\*

Demo on stack using arrays

\*/

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

int bottom=-1;

int top=-1;

int push(int);

int pop();

int stackcont[MAX];

void dispstack();

int push(int v)

{

if(top==MAX-1)

{

printf("\nStackconatiner is full\n");

return 0;

}

top++;

stackcont[top]=v;

return 0;

}

int pop()

{

int v;

if(top==bottom)

{

printf("\nstackcontainer is empty\n");

return 0;

}

v= stackcont[top];

top--;

return v;

}

void dispstack()

{

int iter;

if(top==bottom)

{

printf("\nstackcontainer is empty\n");

return ;

}

printf("\nstack elements are\n");

for(iter=top;iter>bottom;iter--)

{

printf("\n%d",stackcont[iter]);

}

printf("\n\n");

}

int main()

{

int v;

int i;

v=pop();

printf("\n%d element is deleted from the container",v);

push(10);

push(11);

push(12);

push(13);

dispstack();

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

{

v=pop();

printf("\n%d element is deleted from the stack",v);

dispstack();

}

printf("\n\n");

return 0;

}

=>/\*

Demo on stack using Dynamic memory allocation

\*/

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

int bottom=-1;

int top=-1;

int push(int);

int pop();

int \*stackcont;

int size=0;

void dispstack();

void initStack(int maxSize) {

size = maxSize;

stackcont = (int \*)malloc(size \* sizeof(int));

if (stackcont == NULL) {

printf("\nMemory allocation failed!\n");

exit(1);

}

bottom = -1;

top = -1;

}

int push(int v)

{

if(top==size-1)

{

printf("\nStackconatiner is full\n");

return 0;

}

top++;

stackcont[top]=v;

return 0;

}

int pop()

{

int v;

if(top==bottom)

{

printf("\nstackcontainer is empty\n");

return 0;

}

v= stackcont[top];

top--;

return v;

}

void dispstack()

{

int iter;

if(top==bottom)

{

printf("\nstackcontainer is empty\n");

return ;

}

printf("\nstack elements are\n");

for(iter=top;iter>bottom;iter--)

{

printf("\n%d",stackcont[iter]);

}

printf("\n\n");

}

int main()

{

int v;

int i;

initStack(MAX);

\

v=pop();

printf("\n%d element is deleted from the container",v);

push(10);

push(11);

push(12);

push(13);

dispstack();

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

{

v=pop();

printf("\n%d element is deleted from the stack",v);

dispstack();

}

printf("\n\n");

return 0;

}