**1. Introduction**

* 1. **About System :-**

The Graphical Representation of Stack and Queue Represents the Data Structure concepts.

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle.

A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle.

* 1. **About Software Platform Used :**

**INTRODUCTION TO C:**

C is a general-purpose high level language that was originally developed by Dennis Ritchie for the Unix operating system. It was first implemented on the Digital Equipment Corporation PDP-11 computer in 1972.

The Unix operating system and virtually all Unix applications are written in the C language. C has now become a widely used professional language for various reasons..

**Why to use C ?**

C was initially used for system development work, in particular the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

* Operating Systems
* Language Compilers
* Assemblers
* Text Editors
* Print Spoolers
* Network Drivers
* Modern Programs
* Data Bases
* Language Interpreters
* Utilities

C Program File

All the C programs are written into text files with extension ".c" for example *hello.c*. You can use "vi" editor to write your C program into a file.

This tutorial assumes that you know how to edit a text file and how to write programming instructions inside a program file.

A C program basically has the following form:

* Preprocessor Commands
* Functions
* Variables
* Statements & Expressions
* Comments

**Features of C Programming Language :**

**1 .** **Low Level Features :-**

* C Programming provides [low level features](http://www.c4learn.com/c-programming/low-level-language/) that are generally provided by the Lower level languages. C is Closely Related to Lower level Language such as “Assembly Language“.
* It is easier to [write assembly language codes in C programming](http://c4learn.com/c-programs/mixing-inline-assembly-program-in-c.html).

**2 . Portability :-**

* C Programs are portable i.e they can be run on any Compiler with Little or no Modification
* Compiler and Pre-processor make it Possible for C Program to run it on Different PC

**3 . Powerful:-**

* Provides Wide verity of ‘Data Types‘
* Provides Wide verity of ‘Functions’
* Provides useful Control & Loop Control Statements

**4 . Bit Manipulation:-**

* C Programs can be manipulated using bits. We can perform different operations at bit level. We can manage memory representation at bit level. [E.g. [We can use Structure to manage Memory at Bit Level](http://www.c4learn.com/c-programming/c-structure-bit-manipulation/)]
* It provides wide verity of bit manipulation Operators. We have bitwise operators to manage Data at bit level.

**5 . High Level Features :-**

* It is more User friendly as compare to Previous languages. Previous languages such as BCPL, Pascal and other programming languages never provide such great features to manage data.
* Previous languages have their pros and cons but C Programming collected all useful features of previous languages thus C become more effective language.

**6. Efficient Use of Pointers:-**

* Pointers has direct access to memory.
* C Supports efficient use of pointer.

**Disadvantages of C Programming Language:**

* C Programming Language doesn't support Object Oriented Programming (OOP) features like Inheritance, Encapsulation, Polymorphism etc. It is a procedure oriented language. In C, we have to implement any algorithms as a set of function calls.
* C doesn't perform Run Time Type Checking. It only does compile time type checking. At run time, C doesn't ensure whether correct data type is used instead it perform automatic type conversion.
* C does not provides support for namespace like C++. Without Namespace, we cannot declare two variables of same name.
* C doesn't support the concept of constructors and destructors.

**Turbo C features:**

* Inline assembly with full access to the C language symbolic structures and names -- This allowed programmers to write some assembly language codes right into their programs without the need for a separate assembler.
* Support for all memory models -- This had to do with the segmented memory architecture used by 16-bit processors of that era, where each segment was limited to 64 kilobytes (Kb).
* Speed or size optimization -- The compiler could be configured to produce an executable program that was either fast or small in size, but not both.
* Constant folding -- This feature allowed the Turbo C compiler to evaluate constant expressions during compile time rather than during run time.

**1.3 OPERATING ENVIRONMENT - HARDWARE & SOFTWARE**

**HARDWARE:**

* 1 GB Hard Disk
* Minimum 256 RAM
* Key Board : 108 keys
* Monitor :SVGL color

**SOFTWARE:**

1. Operating system : Any Windows-OS

2. Application Software :

Text Editor

Notepad

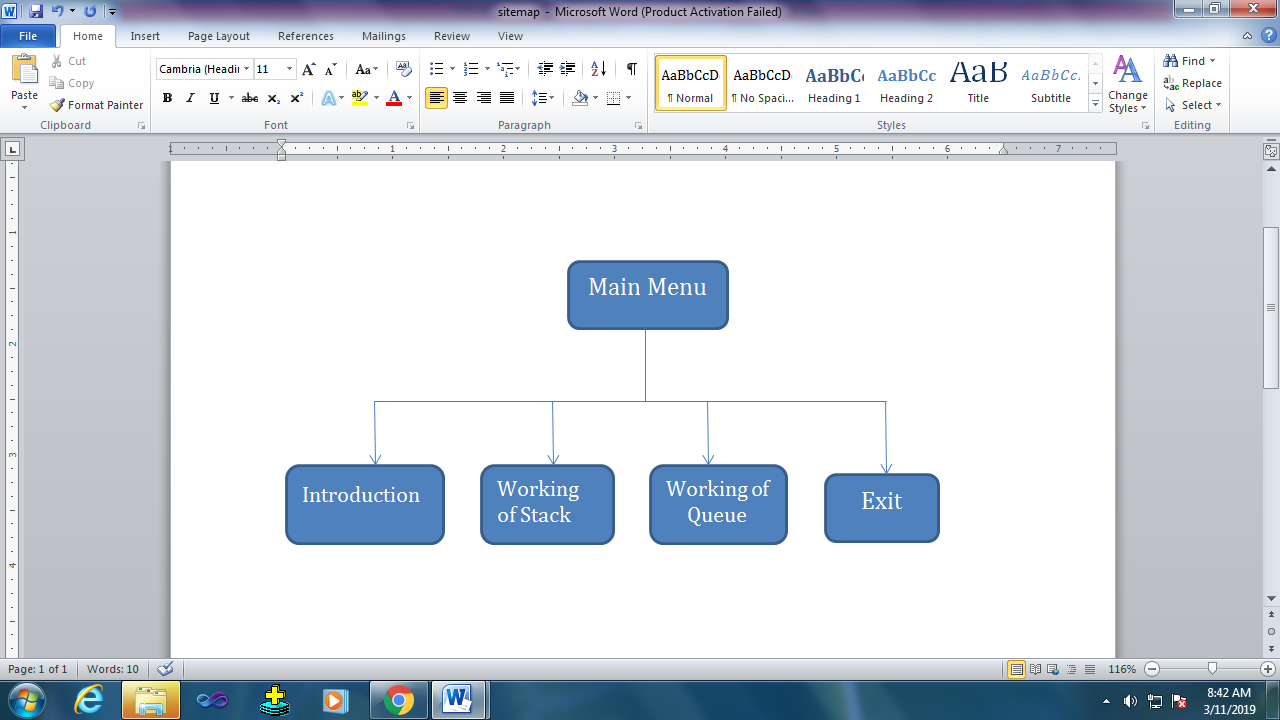
Turbo C++

**2. Analysis**

**2.1 System design (Site Map) :-**

Graphical Representation of

Stack of Queue



**3. Input Design**

1. **Header Files & global Declaration :-**
2. **Main Function**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

#include"intro2.h"

int i,j,ch;

int count=0;

int counts=0;

char n[10];

char elements[10][10];

char qelements[10][10];

void push();

void pop();

void stack();

void insert(); void rem();

void queue();

int front=0;

int rear=0;

void insert()

{

int j=0;

bar(100,50,500,100);

outtextxy(100,50,"Enter any number: ");

gotoxy(50,4);

scanf("%s",qelements[rear++]);

bar(50,30,500,100);

j=60-20\*(count-1);

for(i=0;i<=420-20\*count ;i++)

{

setcolor(0);

bar(j-1+i,150,j+34+i,190);

setcolor(15);

rectangle(j+i,150,j+35+i,190);

outtextxy(j+8+i,165,qelements[rear-1]);

delay(5);

}

}

void rem()

{

int j=0;

j=401-20\*(count-1);

for(i=460;i<=640 ;i++)

{

setcolor(2);

bar(i,150,i+35,190);

setcolor(6);

rectangle(i+1,150,i+36,190);

outtextxy(i+10,160,qelements[front]);

delay(5);

}

for(j=1;j<(rear-front);j++)

{

for(i=460-j\*40;i<=460-(j-1)\*40 ;i++)

{

setcolor(0);

bar(i,150,i+35,190);

setcolor(15);

rectangle(i+1,150,i+36,190);

outtextxy(i+10,160,qelements[front+j]);

delay(5);

}

}

front++;

}

void queue()

{

setcolor(6);

setfillstyle(1,3);//8small rectangle fill colour

setcolor(2); //small r. border colour

rectangle(15,340,500,450); //rectangle size

floodfill(19,350,2);

//setfillstyle(1,9);

//floodfill(150,150,10);

//setfillstyle(1,2);

bar(70,370,150,400);

setcolor(7); //color of text

outtextxy(30,360,"Queue"); //To enter text

setcolor(6);

//outtextxy(220,60,"Enter number:-");

setcolor(4);

//setfillstyle(1,2);

bar(70,370,150,400);

outtextxy(157,380,"1.Insert");

bar(70,370,150,400);

outtextxy(259,382,"2.Remove");

bar(70,370,150,400);

outtextxy(379,382,"3.Back");

setcolor(10); //2line color

line(100,120,500,120);

line(100,225,500,225);

//line(500,220,500,450);

setfillstyle(1,2);

setcolor(6);

setfillstyle(1,0);

j=1;

do

{

bar(100,50,500,100);

outtextxy(100,50,"Enter any choice: ");

gotoxy(50,4);

scanf("%d",&ch);

bar(50,30,500,100);

switch(ch)

{

case 1:if(rear==8)

{

outtextxy(100,50,"Queue Full , can't insert ");

getch();

}

else

{

count++;

insert();

}

break;

case 2:if(count>0)

{

count--;

rem();

}

else

{

// rem();

outtextxy(100,50,"Queue empty, can't delete... ");

getch();

}

break;

case 3: cleardevice();

break;

}

}while(ch!=3);

}

void main()

{

int gd=DETECT,gm;

int n;

int ch,mch;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

//Main menu

do

{

setfillstyle(1,5);//8small rectangle fill colour

setcolor(15); //small r. border colour

floodfill(19,350,1);

setfillstyle(0,1);

outtextxy(120,50,"Graphical Representation of Data Struture");

setcolor(11);

outtextxy(100,100,"Enter your choice:-");

outtextxy(100,130,"1.Introduction");

outtextxy(100,160,"2.Working of Stack");

outtextxy(100,190,"3.Working of Queue");

outtextxy(100,220,"4.Exit");

scanf("%d",&mch); //

clrscr();

switch(mch)

{

case 1:intro();

break;

case 2:cleardevice();

stack();

break;

case 3:cleardevice();

queue();

break;

case 4:break;

}

}while(mch!=4);

}

///////////////////////////////////////////////////////////////Stack//////////////////////////////////////////////////////////

void pop()

{

int j=0;

j=401-20\*(counts-1);

for(i=0;i<=340-20\*counts ;i++)

{

bar(350,j-2-i,475,j+18-i);

setcolor(15);

rectangle(350,j-3-i,430,j+17-i);

outtextxy(375,j+3-i,elements[counts-1]);

delay(5);

}

if(counts>1)

outtextxy(436,j+20,"<-top");

if(counts==1)

bar(350,420,450,422);

bar(325,220,550,140);

}

void push()

{

outtextxy(100,50,"Enter any no.");

gotoxy(30,4);//to input number...

scanf("%s",n);

strcpy(elements[counts],n);

bar(50,30,400,100);

j=counts;

for(i=0;i<=300-20\*j;i++)

{

bar(350,98+i,470,118+i);

setcolor(15);

rectangle(350,100+i,430,120+i);

outtextxy(375,105+i,n);

delay(5);

}

outtextxy(434,100+i,"<-top");

bar(434,120+i,473,140+i);

j++;

}

void stack()

{

setcolor(6);

setfillstyle(1,7);//small rectangle fill colour

setcolor(2); //small r. border colour

rectangle(15,200,200,450); //rectangle size

floodfill(19,210,2);

bar(70,370,150,400);

setcolor(1); //color of text

outtextxy(30,240,"Stack"); //To enter text

setcolor(4);

//outtextxy(220,60,"Enter number:-");

setcolor(9);

//setfillstyle(1,2);

bar(70,370,150,400);

outtextxy(87,300,"1.Push");

bar(70,370,150,400);

outtextxy(89,345,"2.Pop");

bar(70,370,150,400);

outtextxy(89,390,"3.Back");

setcolor(2);

line(330,450,500,450);

line(330,450,330,220);

line(500,220,500,450);

setfillstyle(1,2);

setcolor(6);

setfillstyle(1,0);

j=1;

do

{

bar(100,50,500,100);

outtextxy(100,50,"Enter any choice: ");

gotoxy(50,4);

scanf("%d",&ch);

bar(50,30,500,100);

switch(ch)

{

case 1:if(j==9)

{

outtextxy(100,50,"Overflow , can't push ");

getch();

}

else

{

push();

counts++;

}

break;

case 2:if(counts>0)

{

pop();

counts--;

}

else

{

outtextxy(100,50,"Stack empty, can't pop... ");

getch();

}

break;

case 3: cleardevice();

break;

}

}while(ch!=3);

}

**2.Intro2.h**

void intro()

{

cleardevice();

//clrscr();

settextstyle(8,0,1);

outtextxy(250,50,"..Introduction..");

settextstyle(2,0,4);

printf("\n\n\n\n\n\n\n\nGraphical Representation of Stack and Queue represents the Data");

printf("Structure concepts.");

printf("A stack is a container of objects that are inserted and removed according to the");

printf("to the Last-In-Last-Out (LIFO) principle");

printf("A queue is a container of objects(a linear collection) that are inserted and removed according to the");

printf("to the Last-In-Last-Out (LIFO) principle");

printf("HI");

//clrscr();

printf("dfsF");

getch();

cleardevice();

}

void intro()

{

cleardevice();

//clrscr();

settextstyle(8,0,1);

outtextxy(250,50,"..Introduction..");

settextstyle(2,0,4);

printf("\n\n\n\n\n\n\n\nGraphical Representation of Stack and Queue represents the Data");

printf("Structure concepts.");

printf("A stack is a container of objects that are inserted and removed according to the");

printf("to the Last-In-Last-Out (LIFO) principle");

printf("A queue is a container of objects(a linear collection) that are inserted and removed according to the");

printf("to the Last-In-Last-Out (LIFO) principle");

printf("HI");

//clrscr();

printf("dfsF");

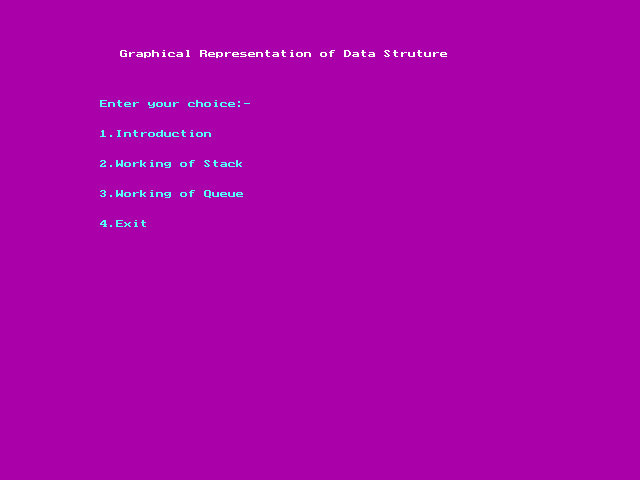
getch();

cleardevice();

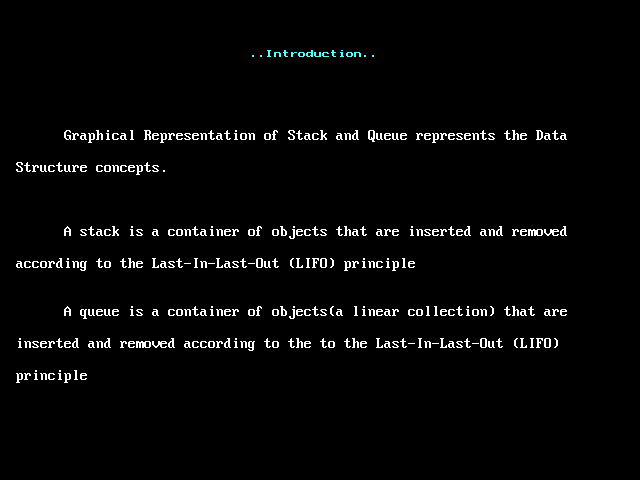
}

**4. Output Design**

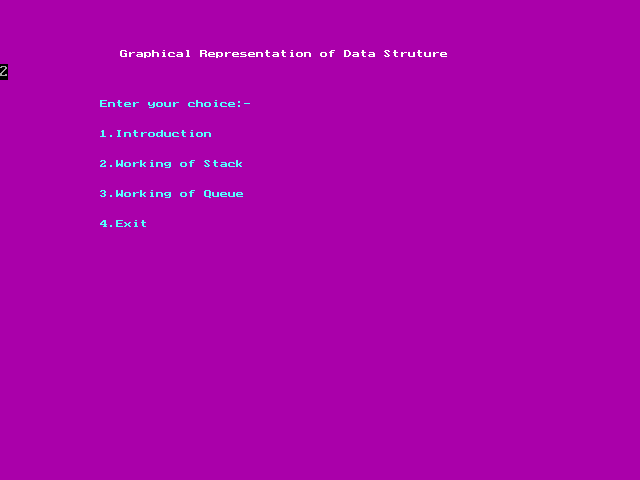
1. **Main Menu :-**



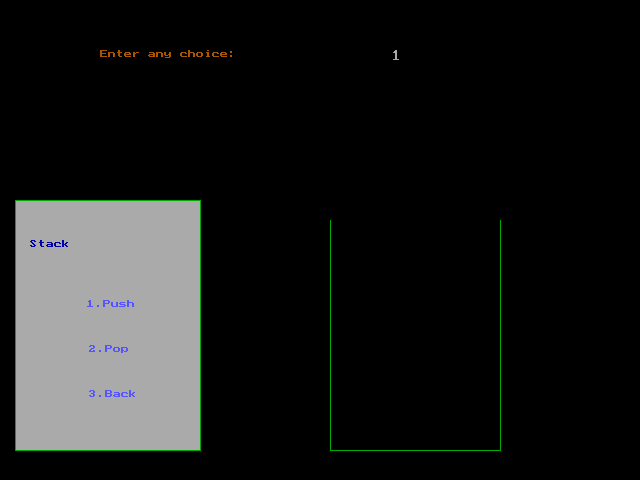
1. **Introduction**



Selecting Stack



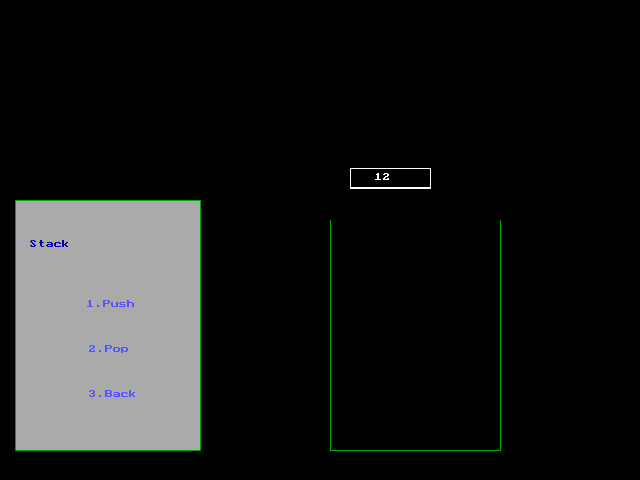
**Entering choice:- (1. Push)**



**Entering Any Number**



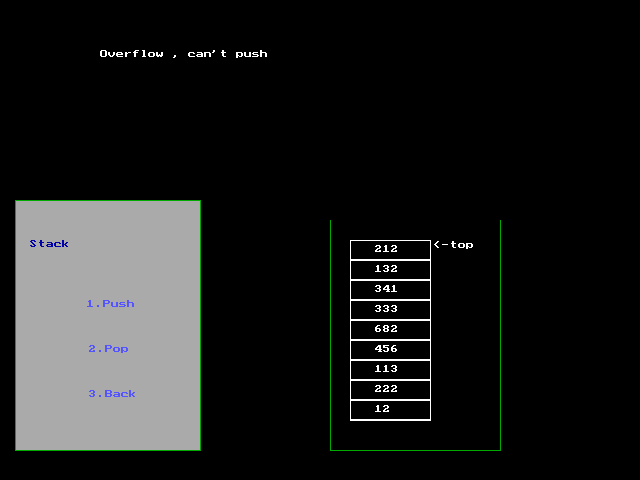
**Value Moving into Stack**



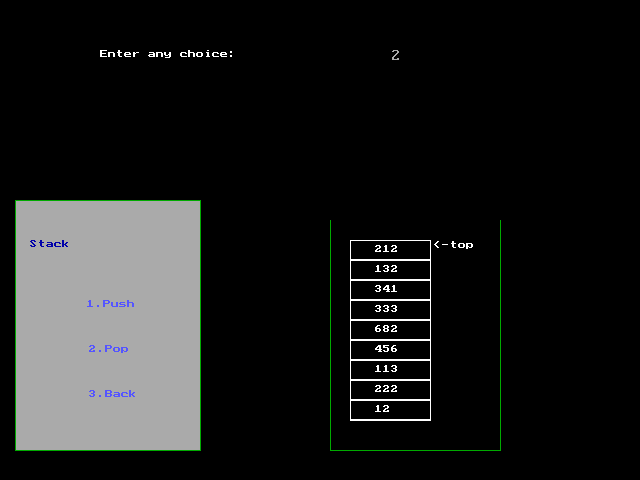
**Value in Stack**



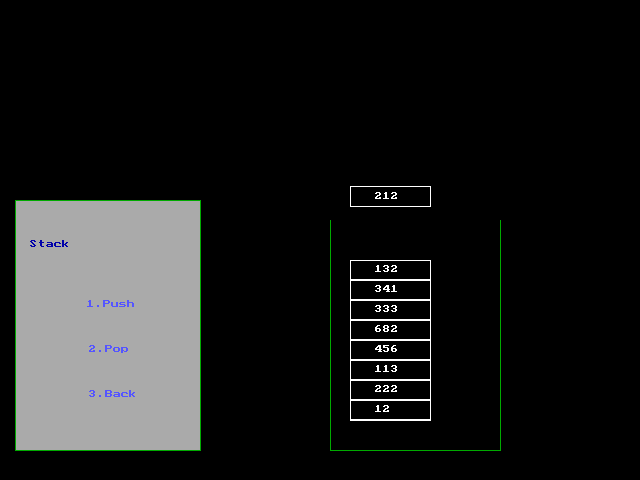
**Stack Overflow**



**Enter Choice :- (2. Pop)**



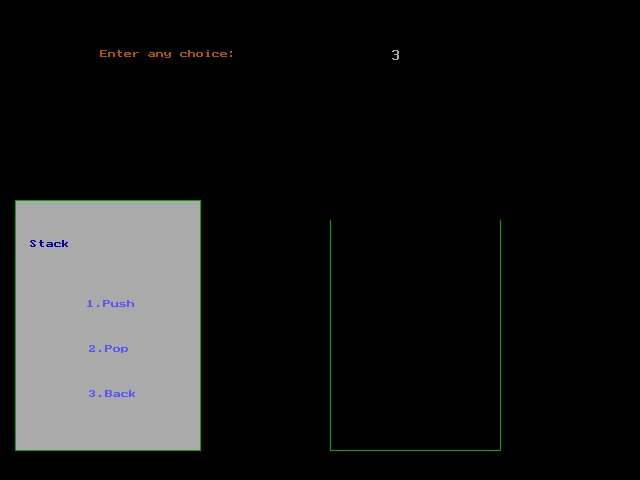
**Element Popped (removed) from Stack**



**Stack Underflow**



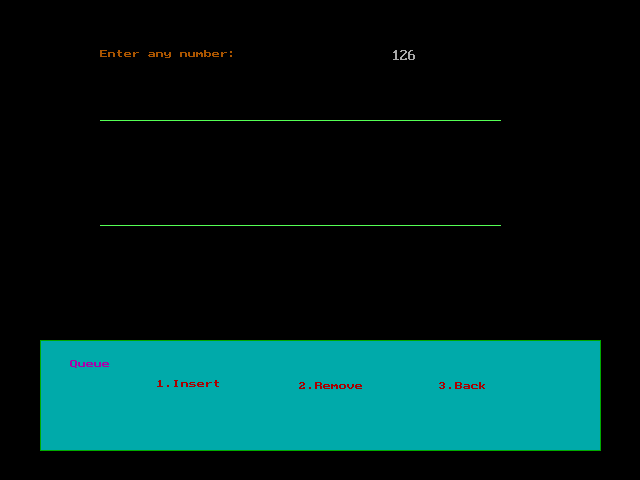
**Choice to go Back**



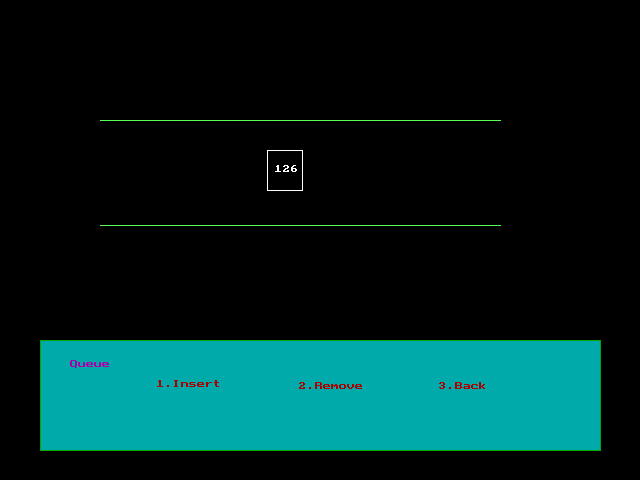
**Enter Choice (1. Insert)**



**Enter Element**



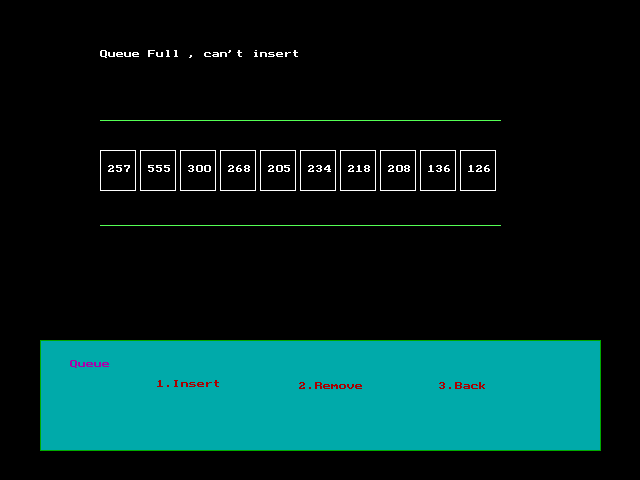
**Element moving into Queue**



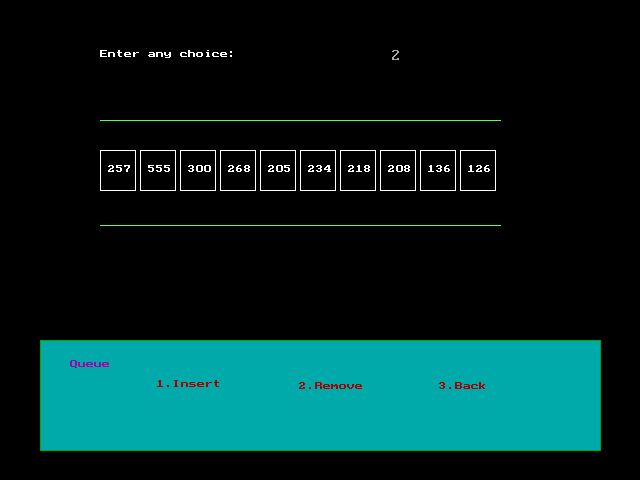
**Element inserted into Queue**



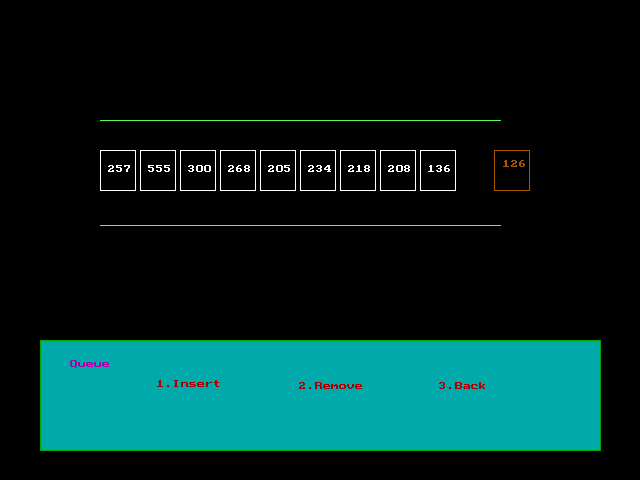
**Condition of Queue - full**



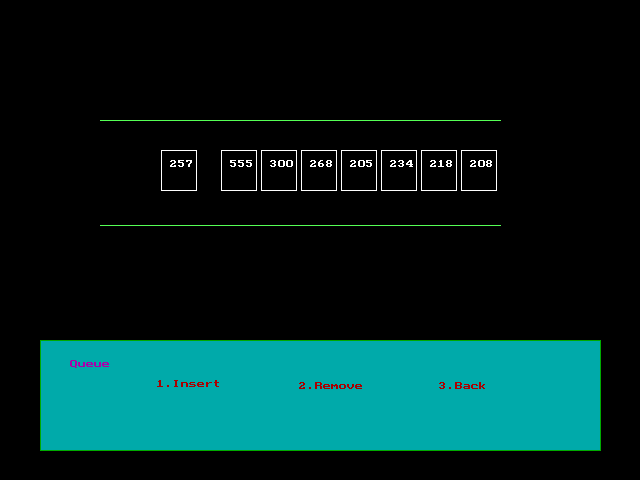
**Choice to Remove Element ( 2. Remove):-**



**Element moving out of Queue**



**Element changing their position**



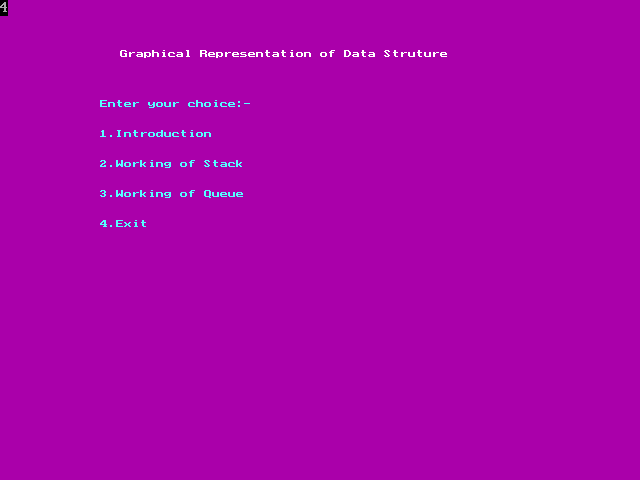
**Queue Underflow**



Choice to go Back



**EXIT.(Enter choice 4 for exit)**



**5. User Manual**

The system is named as Graphical Representation of Stack and Queue. This

System is used to make clear of Stack and Queue.

It contains the following :-

1. **Introduction :-**

It gives a brief introduction of the working of Stack and Queue. Press ‘1’ to get Introduction.

1. **Working of Stack:-**

It shows about how the Stack works. Press ‘2’ to see the graphical working of Stack.

1. **Working of Queue:-**

It shows about how the Queue works. Press ‘3’ to see the graphical working of Queue.

1. **Exit:-**

If you press 4, then you will Exit.

**6. Conclusion**

* We designed the Graphical Representation of Stack of Queue, mainly makes clear the concepts easy.
* Representing the Data Structure concepts had helped us to get knowledge about the working of Stack and Queue.
* In the development of this project we have learned the importance of C as well as graphics related activities.
* We have used Turbo C++ to create this system. Which is used for system development work.

**6. Bibliography**

**Reference Books:**

* **Let Us C**
* Yashwant Kanetkar
* **C Programming Language (2nd Edition)**
* B. W. Kernighan & D. M. Ritchie
* **Project using C**
* PVN. Varalakshmi.
* **Data Structure using C**
* Shilpa Pawale.

**Website**:

* [WWW.w3school.com](http://www.w3school.com/)
* [www.wekipedia.com](http://www.wekipedia.com/)
* [www.google.com](http://www.google.com/) **(Search Engine)**