AJAY KUMAR GARG ENGINEERING COLLEGE, GHAZIABAD

Department of Computer Science and Engineering

Mini Project Report

On

NUMBER SYSTEM CONVERSION TOOL USING STACK DATA STRUCTURE IN C

Submitted By:

Prakhar Tagra

Roll No.:

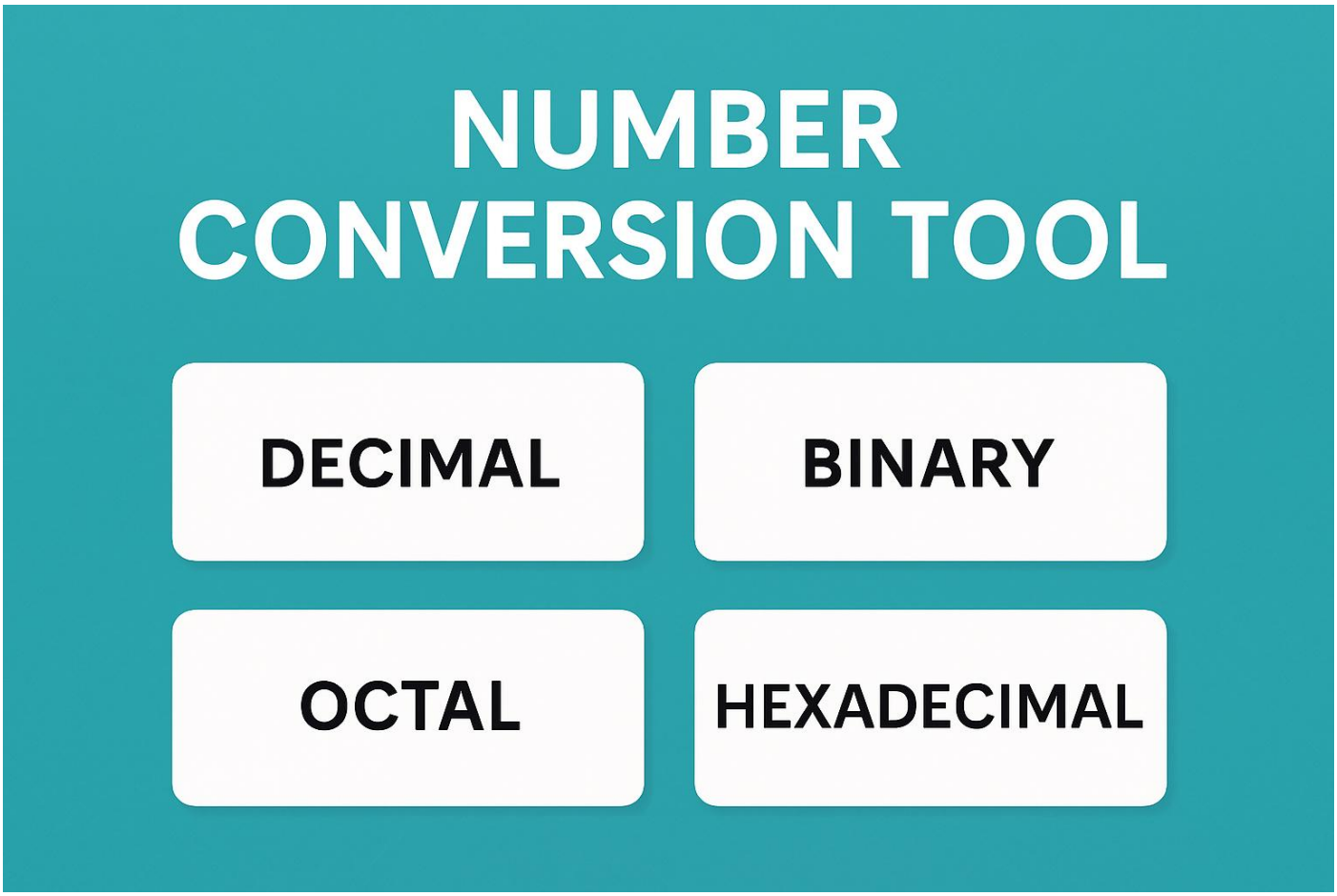
2400270100127

Mentor:

Mr. Vivek Agarwal

Academic Session: 2025-2026

Number System Conversion Using Stack Data Structure in C



Objective:

The objective of this project is to develop a **menu-driven C program** that performs **conversion between different number systems** (Binary, Octal, Decimal, and Hexadecimal) using the **Stack data structure**. The project aims to demonstrate the practical application of stacks in solving real-world problems, particularly in the area of **data representation and conversion in computer systems**.

Problem Statement:

In computer science, different number systems are used for various purposes, such as **binary** for machine-level operations, **octal/hexadecimal** for compact representation, and **decimal** for human interaction. Manually converting between these systems is **time-consuming** and prone to **calculation errors**.

The problem is to **automate** these conversions in a way that:

- Is **fast and accurate**.

- Can handle **large numbers** without manual calculation.
- Supports **both integer and alphanumeric inputs** (for hexadecimal).
- Demonstrates the **LIFO (Last-In-First-Out)** principle of stacks in a practical context.

Scope:

- Decimal → Binary, Octal, Hexadecimal
- Binary → Decimal
- Octal → Decimal
- Hexadecimal → Decimal
- Input handling for **both numeric and string-based numbers**.
- Stack implementation for reversing digits during conversion.

Data Structure Used:

Stack – Implemented using arrays in C.

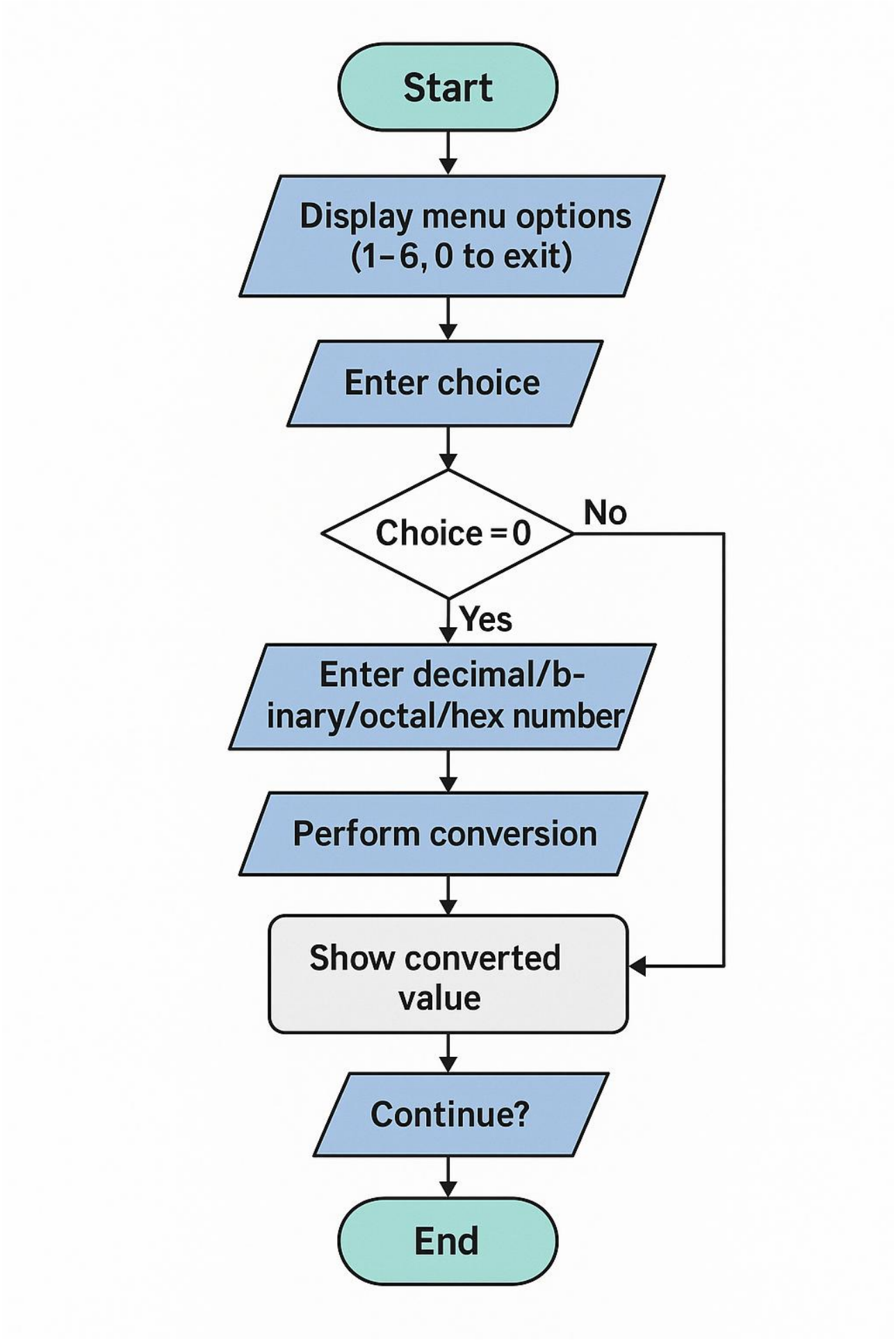
- The stack is used to store intermediate remainders or positional values during number system conversion.
- Follows the **LIFO (Last-In-First-Out)** principle to reverse the order of digits for correct output.

Programming Language Used:

C Language

- Chosen for its **low-level memory control**, **fast execution**, and **simplicity in demonstrating core data structure concepts**.
- Uses standard libraries like:
    - <stdio.h> – for input/output operations.
    - <string.h> – for string length operations in conversions.
    - <math.h> – for power calculations during base-to-decimal conversions.

FLOWCHART:


Start
Display menu options (1–6, 0 to exit)
Enter choice
Choice = 0
No
Yes
Enter decimal/binary/octal/hex number
Perform conversion
Show converted value
Continue?
End

Operations Involved in the Project:

1. **Push Operation**
    - Inserts an element (digit or value) onto the stack during conversion.
2. **Pop Operation**
    - Removes the top element from the stack to retrieve digits in correct order.
3. **Check if Stack is Full (isFull)**
    - Verifies if the stack has reached its maximum capacity before inserting a new element.
4. **Check if Stack is Empty (isEmpty)**
    - Verifies if the stack has no elements before popping.
5. **Decimal to Binary Conversion**
    - Converts a decimal number to its binary representation using repeated division and stack storage.
6. **Decimal to Octal Conversion**
    - Converts a decimal number to octal format.
7. **Decimal to Hexadecimal Conversion**
    - Converts a decimal number to hexadecimal format, including letter digits (A–F).
8. **Binary to Decimal Conversion**
    - Converts a binary number (string input) into decimal using positional values.
9. **Octal to Decimal Conversion**
    - Converts an octal number to decimal.
10. **Hexadecimal to Decimal Conversion**
    - Converts a hexadecimal number (string input with digits and letters) to decimal.
11. **Menu-Driven Execution**
    - Displays a menu of available conversions and executes the chosen operation.
12. **Invalid Input Handling**
    - Detects and displays an error message for invalid base digits.

Operations — explanation & algorithm

Operation: Number Conversion Tool

Implementation Methodology:

Use arrays, stack implementation, and functions to convert numbers between different bases (Decimal, Binary, Octal, Hexadecimal). The program takes user input at runtime, processes it using push/pop stack operations, and displays the converted value.

Variable	Data Type / Structure
numstack	int array
MAXSIZE	int
top	int
number	int
base	int
choice	int
digit	int
strnum	char array
value	int
i, j	int
isfull()	Function returning int
isempty()	Function returning int
push()	Function returning void
pop()	Function returning int
numberConversion1()	Function returning void
numberConversion2()	Function returning void

Operation: isfull()

Details: Tests whether the stack array (numstack) is full to prevent overflow.

Implementation methodology:

Compare global top with MAXSIZE - 1. Returns 1 if full, else 0.

Algorithm (steps):

1. Read global top.
2. If top == MAXSIZE - 1 → return 1 (stack full).
3. Else → return 0 (not full).
   Time complexity: O(1)

```
int isfull(){
    if(top==MAXSIZE-1){
        return 1;
    }
    else{
        return 0;
    }
}
```

Operation: isempty()

Details: Tests whether the stack is empty to prevent underflow.

Implementation methodology:

Checks if top equals -1. Returns 1 if empty, else 0.

Algorithm (steps):

1. Read global top.
2. If top == -1 → return 1 (stack empty).
3. Else → return 0 (not empty).
   Time complexity: O(1)

```
int isempty(){
    if(top==-1){
        return 1;
    }
    else{
        return 0;
    }
}
```

Operation: push(int data)

Details: Pushes an integer data onto numstack (used to store remainders or partial values).

Implementation methodology:

Calls isfull(); if not full, increments top then stores numstack[top]=data. If full, prints an overflow message. Void function (no status returned).

Algorithm (steps):

1. Call isfull().
2. If it returns 1 → print overflow message and return.
3. Else → do top = top + 1.
4. Store numstack[top] = data.
5. Return.
   Time complexity: O(1)

```
void push(int data){
    if(!isfull()){
        top=top+1;
        numstack[top]=data;
    }
    else{
        printf("\nSTACK IS FULL....NO OTHER ELEMENTS CAN BE ADDED!!!");
    }
}
```

Operation: pop()

Details: Pops and returns the top integer from numstack.

Implementation methodology:

Calls isempty(); if not empty, reads numstack[top], decrements top, returns the value. If empty, prints a message and returns 0.

Algorithm (steps):

1. Call isempty().
2. If it returns 1 → print underflow message and return 0.
3. Else → data = numstack[top].
4. top = top - 1.

5. return data.
   Time complexity: O(1)

```
int pop(){
    if(!isempty()){
        int data;
        data = numstack[top];
        top = top-1;
        return data;
    }
    else{
        printf("\nSTACK IS EMPTY..");
        return 0;
    }
}
```

Operation: numberConversion1

Details: Converts an integer number (decimal) into digits of base by repeated division; pushes remainders onto the stack.

Implementation methodology:

While number != 0, compute number % base and push(remainder); then number = number / base. Does not handle number == 0 or negative numbers explicitly inside this function.

Algorithm (steps):

1. Input: integer number, integer base.
2. While number != 0:
   a. remainder = number % base.
   b. push(remainder).
   c. number = number / base.
3. Return to caller (caller pops and prints digits).
   Time complexity: O(k) where k = number of digits in the target base.

```
void numberConversion1(int number, int base){
    while(number!=0){
        push(number%base);
        number=number/base;
    }
}
```

Operation: numberConversion2

Details: Converts a string number representing digits in base into decimal by iterating the string right-to-left, computing each digit value, computing value*pow(base,i) and *pushing* that product to stack. Your main sums popped values to get decimal result. Accepts 0-9, A-F, a-f. Prints "Invalid Input!!" and returns if a character is invalid.

Implementation methodology:

- Iterate j from strlen(number)-1 down to 0.
- For each character derive value (0–15).
- Compute value * pow(base, i) and push(...).
- Increment i (power index). Caller later pops all pushed products and sums them to get decimal.

Algorithm (steps):

1. Let i = 0.
2. For j = strlen(number)-1 down to 0:
   a. If number[j] in '0'..'9' → value = number[j] - '0'.
   b. Else if 'A'..'F' → value = number[j] - 'A' + 10.
   c. Else if 'a'..'f' → value = number[j] - 'a' + 10.
   d. Else → print "Invalid Input!!" and return.
   e. Compute term = value * pow(base, i).
   f. push(term).
   g. i = i + 1.
3. Return to caller (caller pops and sums all pushed term values to form decimal result).
   Time complexity: O(n) where n = number of digits in input string.

```
void numberConversion2(char number[100], int base){
    int value,i=0,j;
    for(j=strlen(number)-1;j>=0;j--){
        if(number[j]>='0' && number[j]<='9'){
            value = number[j]-'0';
        }
        else if(number[j]>='A' && number[j]<='F'){
            value = number[j]-'A'+10;
        }
        else if(number[j]>='a' && number[j]<='f'){
            value = number[j]-'a'+10;
        }
        else{
            printf("\nInvalid Input!!");
            return;
            }
        push(value*pow(base,i));
        i++;
    }
}
```

Operation: main() (menu-driven execution)

Details: Presents a menu, reads user choice, resets top=-1 at the start of each loop, calls the appropriate conversion function, and prints results. Handles 6 conversion choices + exit. Uses scanf for inputs. For decimal→base it calls numberConversion1(...) then pops to print digits; for base→decimal it calls numberConversion2(...) then pops all pushed place-values and sums them to print decimal.

Implementation methodology:

- Infinite while(1) loop; inside, set top = -1 to reset stack for new operation.
- Display menu, scanf("%d", &choice).
- switch(choice) to dispatch:
    - Cases 1–3: prompt for decimal number, call numberConversion1(number, base), then while not isempty() pop digits and print (mapping >=10 to 'A'..'F').
    - Cases 4–6: prompt for string strnum, call numberConversion2(strnum, base), then pop all terms, accumulate into digit and print digit.
    - Case 0: exit.

Algorithm (main loop steps):

1. Loop start: top = -1.
2. Display menu and read choice.
3. If choice == 0 → print exit message and break.
4. switch(choice):
    - If decimal→base (1/2/3):
      a. Read integer number.
      b. Call numberConversion1(number, target_base).
      c. Initialize accumulator/display. While !isempty(): pop() and print digits (convert 10→'A', etc.).
    - If base→decimal (4/5/6):
      a. Read string strnum.
      b. Call numberConversion2(strnum, target_base).
      c. digit = 0. While !isempty(): digit += pop(). Print digit.
    - Default: print "NO SUCH CHOICE FOUND".

```
**********MENU CARD**********
**************************************************
Choose The Required Option From Following:
1. Decimal To Binary:
2. Decimal To Octal:
3. Decimal To HexaDecimal:
4. Binary To Decimal:
5. Octal To Decimal:
6. HexaDecimal To Decimal:
0. Exit the Code
**************************************************

**************************************************
Enter Your Choice: 6

**************************************************
Enter Hexadecimal Number: 1abc5

Converted Answer in Decimal: 109509
**********MENU CARD**********
**************************************************
Choose The Required Option From Following:
1. Decimal To Binary:
2. Decimal To Octal:
3. Decimal To HexaDecimal:
4. Binary To Decimal:
5. Octal To Decimal:
6. HexaDecimal To Decimal:
0. Exit the Code
**************************************************

**************************************************
Enter Your Choice:
```

FINAL WORKING CODE:

```
#include<stdio.h>
#include<string.h>
#include<math.h>
int numstack[100];
int MAXSIZE = 100;
int top = -1;

int isfull(){
   if(top==MAXSIZE-1){
      return 1;
   }
   else{
      return 0;
   }
}

int isempty(){
   if(top==-1){
      return 1;
   }
   else{
      return 0;
   }
}

void push(int data){
   if(!isfull()){
      top=top+1;
      numstack[top]=data;
   }
   else{
      printf("\nSTACK IS FULL....NO OTHER ELEMENTS CAN BE ADDED!!!");
   }
}

int pop(){
   if(!isempty()){
      int data;
      data = numstack[top];
      top = top-1;
      return data;
   }
   else{
```

```
        printf("\nSTACK IS EMPTY..");
        return 0;
    }
}

void numberConversion1(int number, int base){
    while(number!=0){
        push(number%base);
        number=number/base;
    }
}

void numberConversion2(char number[100], int base){
    int value,i=0,j;
    for(j=strlen(number)-1;j>=0;j--){
        if(number[j]>='0' && number[j]<='9'){
            value = number[j]-'0';
        }
        else if(number[j]>='A' && number[j]<='F'){
            value = number[j]-'A'+10;
        }
        else if(number[j]>='a' && number[j]<='f'){
            value = number[j]-'a'+10;
        }
        else{
            printf("\nInvalid Input!!");
            return;
            }
        push(value*pow(base,i));
        i++;
    }
}

int main(){
        int base,choice,number,digit=0,i;
        char strnum[100];
        while(1){
        top=-1;
        printf("\n**********MENU CARD**********\n");
        for(i=0;i<=50;i++){
            printf("*");
        }
        printf("\nChoose The Required Option From Following: ");
        printf("\n1. Decimal To Binary: ");
```

```
    printf("\n2. Decimal To Octal: ");
    printf("\n3. Decimal To HexaDecimal: ");
    printf("\n4. Binary To Decimal: ");
    printf("\n5. Octal To Decimal: ");
    printf("\n6. HexaDecimal To Decimal: ");
    printf("\n0. Exit the Code");
    printf("\n");
    for(i=0;i<=50;i++){
       printf("*");
    }
    printf("\n");
    printf("\n");
    for(i=0;i<=50;i++){
       printf("*");
    }
    printf("\nEnter Your Choice: ");
    scanf("%d",&choice);
    printf("\n");
    for(i=0;i<=50;i++){
       printf("*");
    }
    if(choice == 0){
       printf("\nExiting the Program... GoodBye!!");
       break;
    }
    switch(choice){
       case 1:
       printf("\nEnter Decimal Number: ");
       scanf("%d",&number);
       numberConversion1(number,base=2);
       printf("\nConverted Answer in Binary: ");
       digit=0;
       while(!isempty()){
          digit = pop();
          if(digit>=10){
             digit = 'A'+(digit-10);
             printf("%c",digit);
          }
          else{
             printf("%d",digit);
          }
       }
       break;
       case 2:
```

```
numberConversion1(number,base=8);
printf("\nConverted Answer in Octal: ");
digit=0;
while(!isempty()){
   digit = pop();
   if(digit>=10){
      digit = 'A'+(digit-10);
      printf("%c",digit);
   }
   else{
      printf("%d",digit);
   }
}
break;
case 3:
printf("\nEnter Decimal Number: ");
scanf("%d",&number);
numberConversion1(number,base=16);
printf("\nConverted Answer in HexaDecimal: ");
digit=0;
while(!isempty()){
   digit = pop();
   if(digit>=10){
      digit = 'A'+(digit-10);
      printf("%c",digit);
   }
   else{
      printf("%d",digit);
   }
}
break;
case 4:
printf("\nEnter Binary Number: ");
scanf("%s",strnum);
numberConversion2(strnum,base=2);
printf("\nConverted Answer in Decimal: ");
digit=0;
while(!isempty()){
   digit = digit + pop();
}
   printf("%d",digit);
break;
```

```
        case 5:
        printf("\nEnter Octal Number: ");
        scanf("%s",strnum);
        numberConversion2(strnum,base=8);
        printf("\nConverted Answer in Decimal: ");
        digit=0;
        while(!isempty()){
           digit = digit + pop();
        }
           printf("%d",digit);
        break;
        case 6:
        printf("\nEnter Hexadecimal Number: ");
        scanf("%s",strnum);
        numberConversion2(strnum,base=16);
        printf("\nConverted Answer in Decimal: ");
        digit=0;
        while(!isempty()){
           digit = digit + pop();
        }
           printf("%d",digit);
        break;
        default:
        printf("\n**NO SUCH CHOICE FOUND**");
     }
   }
}
```

****