#include<stdio.h>

#define SIZE 100

struct student

{

int id;

int age;

int marks;

};

int main()

{

struct student s[SIZE];

int i,n;

printf("Please enter the number of students:");

scanf("%d",&n);

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

{

printf("Enter the student id:\n");

scanf("%d",&s[i].id);

printf("Enter the age of the student:\n");

scanf("%d",&s[i].age);

printf("Enter the marks of the student:\n");

scanf("%d",&s[i].marks);

}

printf("The details of student(s) who are qualified:\n");

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

{

if(s[i].age>20 && s[i].marks>=65 && s[i].marks<=100)

{

printf("Student ID:%d\n",s[i].id);

printf("Student age:%d\n",s[i].age);

printf("Student marks:%d\n",s[i].marks);

}

}

return 0;

}

Text

Description automatically generated

#include<stdio.h>

#include<stdlib.h>

#define SIZE 10

void push(int);

void pop();

void display();

int stack[SIZE],top=-1;

int main()

{

int n,choice;

while(1)

{

printf("\nMENU\n\n");

printf("(1)Push\n");

printf("(2)Pop\n");

printf("(3)Display\n");

printf("(4)Exit\n");

printf("Enter your choice: \n\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the value to be inserted:");

scanf("%d",&n);

push(n);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

default: printf("Incorrect Selection.Select Again!\n\n");

}

}

return 0;

}

void push(int n)

{

if(top==SIZE-1)

{

printf("Stack is Full.Insertion is not possible!\n\n");

}

else

{

top++;

stack[top]=n;

printf("Insertion Successful\n\n");

}

}

void pop()

{

if(top==-1)

{

printf("Stack is empty.Deletion is not possible!\n\n");

}

else

{

printf("Deleted: %d\n\n",stack[top]);

top--;

}

}

void display()

{

if(top==-1)

{

printf("Stack is Empty\n\n");

}

else

{

int i;

printf("Stack elemrnts are: \n\n");

for(i=top;i>=0;i--)

printf("%d\n",stack[i]);

}

}Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated

#include<stdio.h>

#include<ctype.h>

#define SIZE 50

char stack[SIZE];

int top=-1;

void push(char elem)

{

stack[++top]=elem;

}

char pop()

{

return stack[top--];

}

int pr(char symbol)

{

if(symbol=='^')

{

return(3);

}

else if(symbol=='\*'||symbol=='/')

{

return(2);

}

else if(symbol=='+'||symbol=='-')

{

return(1);

}

else

{

return(0);

}

}

int main()

{

char infix[50],postfix[50],ch,elem;

int i=0,k=0;

printf("Enter the Infix expression: ");

scanf("%s",&infix);

push('#');

while((ch=infix[i++])!='\0')

{

if(ch=='(')push(ch);

else

if(isalnum(ch))postfix[k++]=ch;

else

if(ch==')')

{

while(stack[top]!='(')

postfix[k++]=pop();

elem=pop();

}

else

{

while(pr(stack[top])>=pr(ch))

postfix[k++]=pop();

push(ch);

}

}

while(stack[top]!='#')

postfix[k++]=pop();

postfix[k]='\0';

printf("\nPostfix Expression = %s\n",postfix);

return 0;

}#include<stdio.h>

#include<ctype.h>

#define SIZE 50

char stack[SIZE];

int top=-1;

void push(char elem)

{

stack[++top]=elem;

}

char pop()

{

return stack[top--];

}

int pr(char symbol)

{

if(symbol=='^')

{

return(3);

}

else if(symbol=='\*'||symbol=='/')

{

return(2);

}

else if(symbol=='+'||symbol=='-')

{

return(1);

}

else

{

return(0);

}

}

int main()

{

char infix[50],postfix[50],ch,elem;

int i=0,k=0;

printf("Enter the Infix expression: ");

scanf("%s",&infix);

push('#');

while((ch=infix[i++])!='\0')

{

if(ch=='(')push(ch);

else

if(isalnum(ch))postfix[k++]=ch;

else

if(ch==')')

{

while(stack[top]!='(')

postfix[k++]=pop();

elem=pop();

}

else

{

while(pr(stack[top])>=pr(ch))

postfix[k++]=pop();

push(ch);

}

}

while(stack[top]!='#')

postfix[k++]=pop();

postfix[k]='\0';

printf("\nPostfix Expression = %s\n",postfix);

return 0;

}

Text

Description automatically generated

#include<stdio.h>

#include<stdlib.h>

#define MAX 50

int queue[MAX];

int front = -1;

int rear = -1;

void insert()

{

int element;

if(rear==MAX-1)

printf("Queue Overflow!\n");

else

{

if(front==-1)

front=0;

printf("Enter the element to be inserted into the Queue: ");

scanf("%d",&element);

queue[++rear]=element;

printf("Element successfully inserted!!!\n");

}

}

void delete()

{

if(front==-1||front>rear)

{

printf("Queue underflow!\n");

}

else

{

printf("Deleted Element: %d\n",queue[front++]);

if(front>rear)

{

front=-1;

rear=-1;

}

}

}

void display()

{

int i;

if(front==-1)

{

printf("Queue is empty!\n");

}

else

{

printf("Elements in the Queue are : \n");

for(i=front;i<=rear;i++)

printf("\n%d",queue[i]);

}

}

int main()

{

int choice;

while(1)

{

printf("\n\n");

printf("(1)INSERT\n(2)DELETE\n(3)DISPLAY\n(4)EXIT");

printf("\nEnter you choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

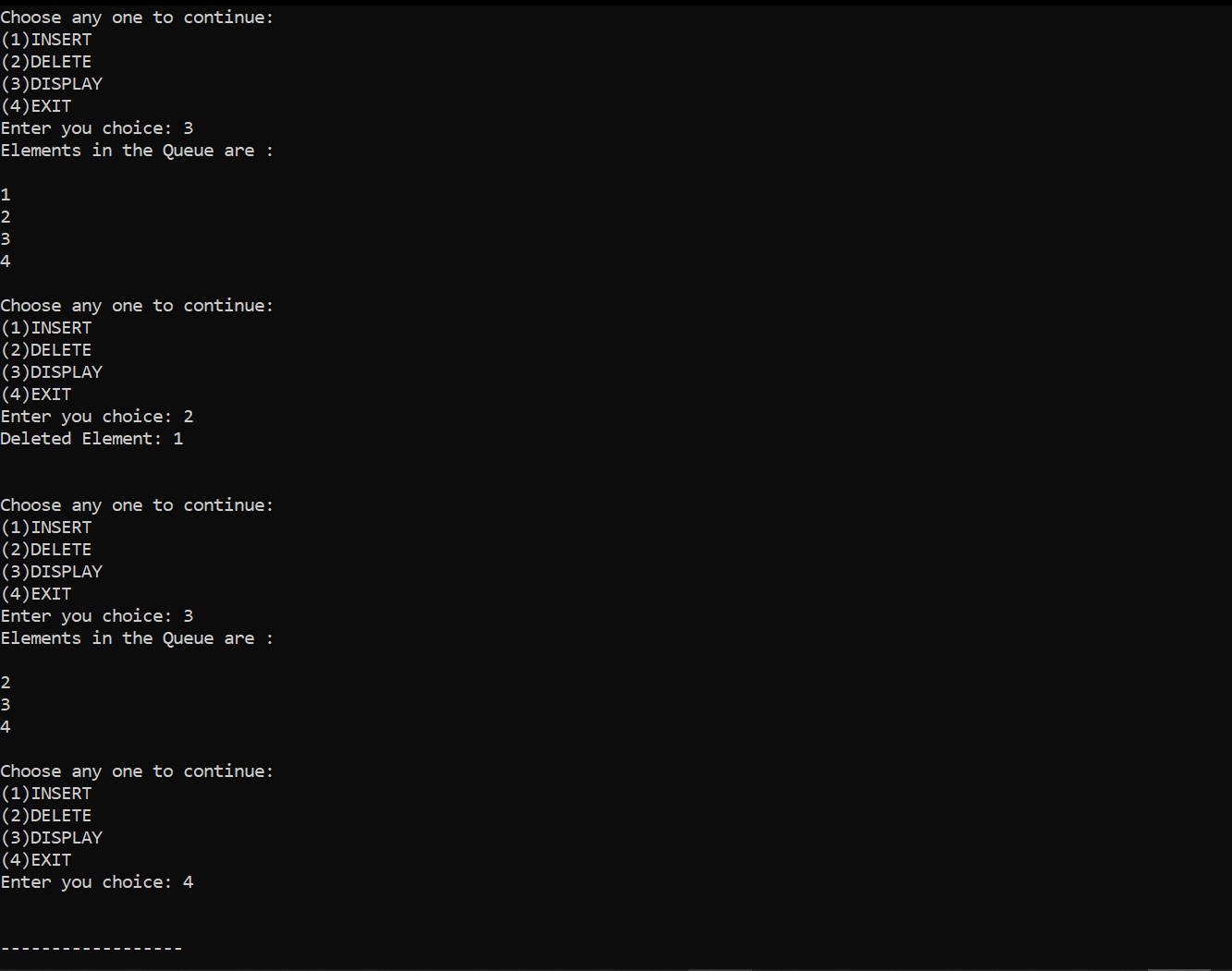
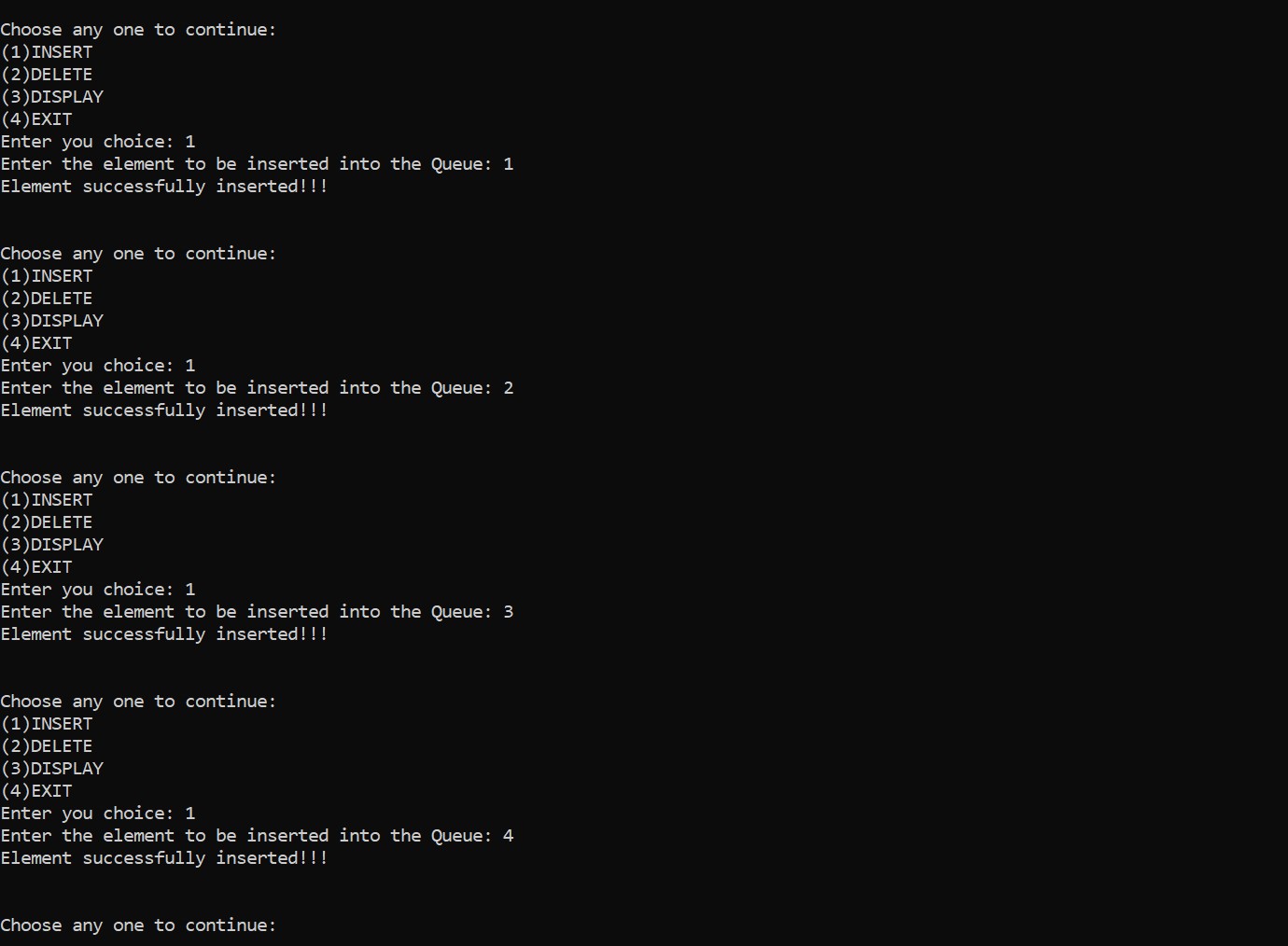
printf("Invalid choice!\n");

}

}

return 0;

}



#include<stdio.h>

#include<stdlib.h>

#define size 5

int Q[size];

int front=-1,rear=-1;

int isfull()

{

if(front==rear+1||(front==0&&rear==size-1))

{

return 1;

}

return 0;

}

int isempty()

{

if(front==-1)

{

return 1;

}

return 0;

}

void enqueue()

{

int element;

if(isfull())

{

printf("\nQueue Overflow!!!\n");

}

else

{

if(front==-1)

{

front=0;

}

printf("\nEnter the element to be inserted into the Queue: ");

scanf("%d",&element);

rear = (rear+1)%size;

Q[rear]=element;

printf("\nInserted element is %d\n",element);

}

}

void dequeue()

{

int element;

if(isempty())

{

printf("\nQueue Underflow!\n");

}

else

{

element=Q[front];

if(front==rear)

{

front=-1;

rear=-1;

}

else

{

front=(front+1)%size;

}

printf("\nDeleted Element is %d\n",element);

}

}

void display()

{

int i;

if(isempty())

{

printf("\nQueue is Empty!!Enter some Elements!!\n");

}

else

{

printf("\nFront--> %d",front);

printf("\nQueue Elements: \n");

for(i=front;i!=rear;i=(i+1)%size)

{

printf("%d\n",Q[i]);

}

printf("%d",Q[i]);

printf("\nRear--> %d\n",rear);

}

}

int main()

{

int choice;

while(1)

{

printf("\n-----MENU-----\n");

printf("(1)INSERT\n");

printf("(2)DELETE\n");

printf("(3)DISPLAY\n");

printf("(4)EXIT\n");

printf("Enter you choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

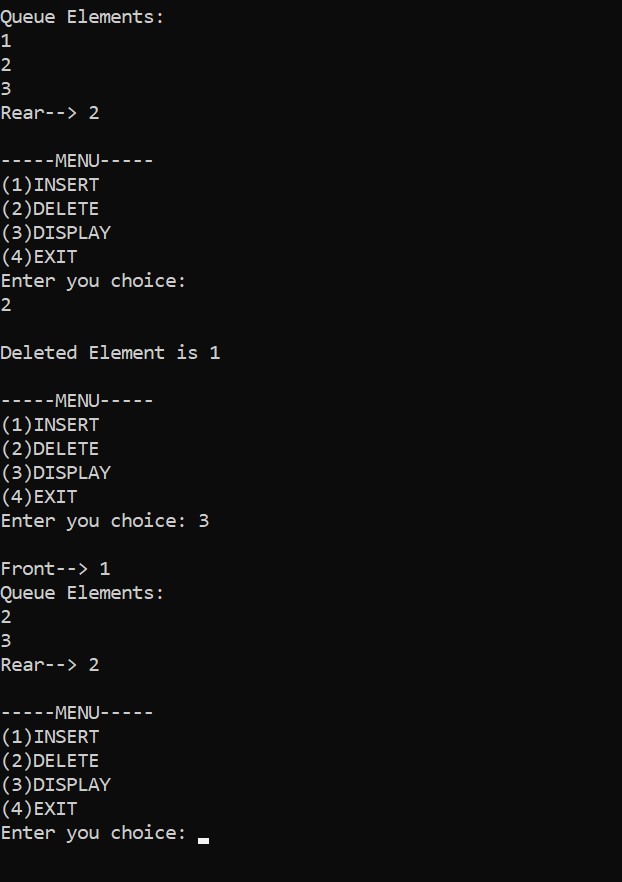
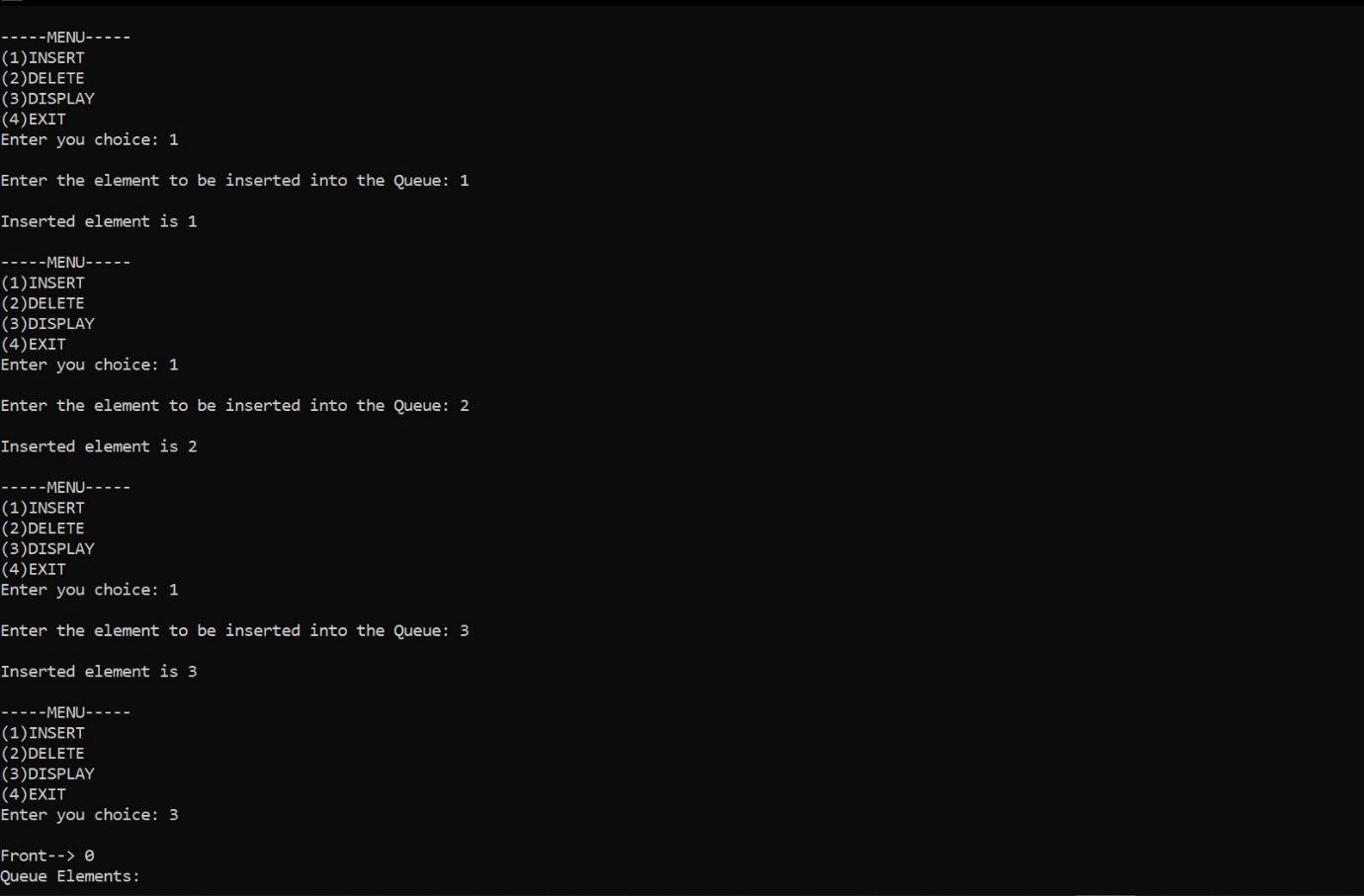
printf("\nInvalid choice!!Choose another one\n");

}

}

return 0;

}



#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void insert\_beg();

void insert\_last();

void insert\_random();

void delete\_beg();

void delete\_last();

void delete\_random();

void display();

int item;

void main(){

int choice =0;

while(1){

printf("\*\*\*\*\*\*\*///MENU\\\\*\*\*\*\*\*\*\*\n\n");

printf("Choose your option::>\n");

printf("[1]Insert at Begining\n\n");

printf("[2]Insert at Last\n\n");

printf("[3]Insert at Random Location\n\n");

printf("[4]Delete at Begining\n\n");

printf("[5]Delete at Last\n\n");

printf("[6]Delete at Random Location\n\n");

printf("[7]Display\n\n");

printf("[8]Exit..\n\n");

printf("Enter your option::>\n");

scanf("%d", &choice);

switch(choice){

case 1:

insert\_beg();

break;

case 2:

insert\_last();

break;

case 3:

insert\_random();

break;

case 4:

delete\_beg();

break;

case 5:

delete\_last();

break;

case 6:

delete\_random();

break;

case 7:

display();

break;

case 8:

exit(0);

break;

default: printf("\nInvalid Option!!!\n");

}

}

}

void insert\_beg(){

struct node \*ptr;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr==NULL){

printf("\nOVERFLOW!!!\n");

}

else{

printf("Enter the value of node::>\n");

scanf("%d", &item);

ptr->data = item;

ptr->next = head;

head = ptr;

printf("Node has been Successfully Inserted!!\n");

}

}

void insert\_last(){

struct node \*ptr, \*temp;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr==NULL){

printf("\nOVERFLOW!!!\n");

}

else{

printf("Enter Value of Node::>\n");

scanf("%d", &item);

ptr->data = item;

if(head==NULL){

ptr->next = NULL;

head = ptr;

printf("Node Successfully Inserted!!\n");

}

else{

temp = head;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = ptr;

ptr->next = NULL;

printf("Node Successfully Inserted!!\n");

}

}

}

void insert\_random(){

int locat;

struct node \*ptr, \*temp;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr==NULL){

printf("\nOVERFLOW!!!\n");

}

else{

printf("Enter the value of the Node::>\n");

scanf("%d", &item);

ptr->data = item;

printf("Enter the location to which you want the element to be inserted::>\n");

scanf("%d", &locat);

temp = head;

if(locat==1){

ptr->next = temp;

head = ptr;

return;

}

for(int i=0;i<locat-1;i++){

temp = temp->next;

if(temp==NULL){

printf("\nInsertion Failed!!\n");

return;

}

}

ptr->next = temp->next;

temp->next = ptr;

printf("Insertion Successfull!!\n");

}

}

void delete\_beg(){

struct node \*ptr;

if(head==NULL){

printf("\nLIST IS EMPTY!!!\n");

}

else{

ptr = head;

head = ptr->next;

free(ptr);

printf("Node deleted from Begining!!\n");

}

}

void delete\_last(){

struct node \*ptr, \*ptr1;

if(head==NULL){

printf("\nLIST IS EMPTY!!!\n");

}

else if(head->next==NULL){

head = NULL;

free(head);

printf("Only Node Of List Deleted!!!\n");

}

else{

ptr = head;

while(ptr->next!=NULL){

ptr1 = ptr;

ptr = ptr->next;

}

ptr1->next = NULL;

free(ptr);

printf("\nDeleted Node from Last!!!\n");

}

}

void delete\_random(){

struct node \*ptr, \*ptr1;

int locat;

printf("\nEnter the location of the Node to be Deleted::>\n");

scanf("%d", &locat);

ptr = head;

for(int i=0;i<locat;i++){

ptr1 = ptr;

ptr = ptr->next;

if(ptr==NULL){

printf("Cannot Delete!!\n");

}

}

ptr1->next = ptr->next;

free(ptr);

printf("\nDELETED NODE::> %d", locat+1);

}

void display(){

struct node \*ptr;

ptr = head;

if(ptr==NULL){

printf("\nEMPTY!!!\n");

}

else{

printf("\n");

while(ptr!=NULL){

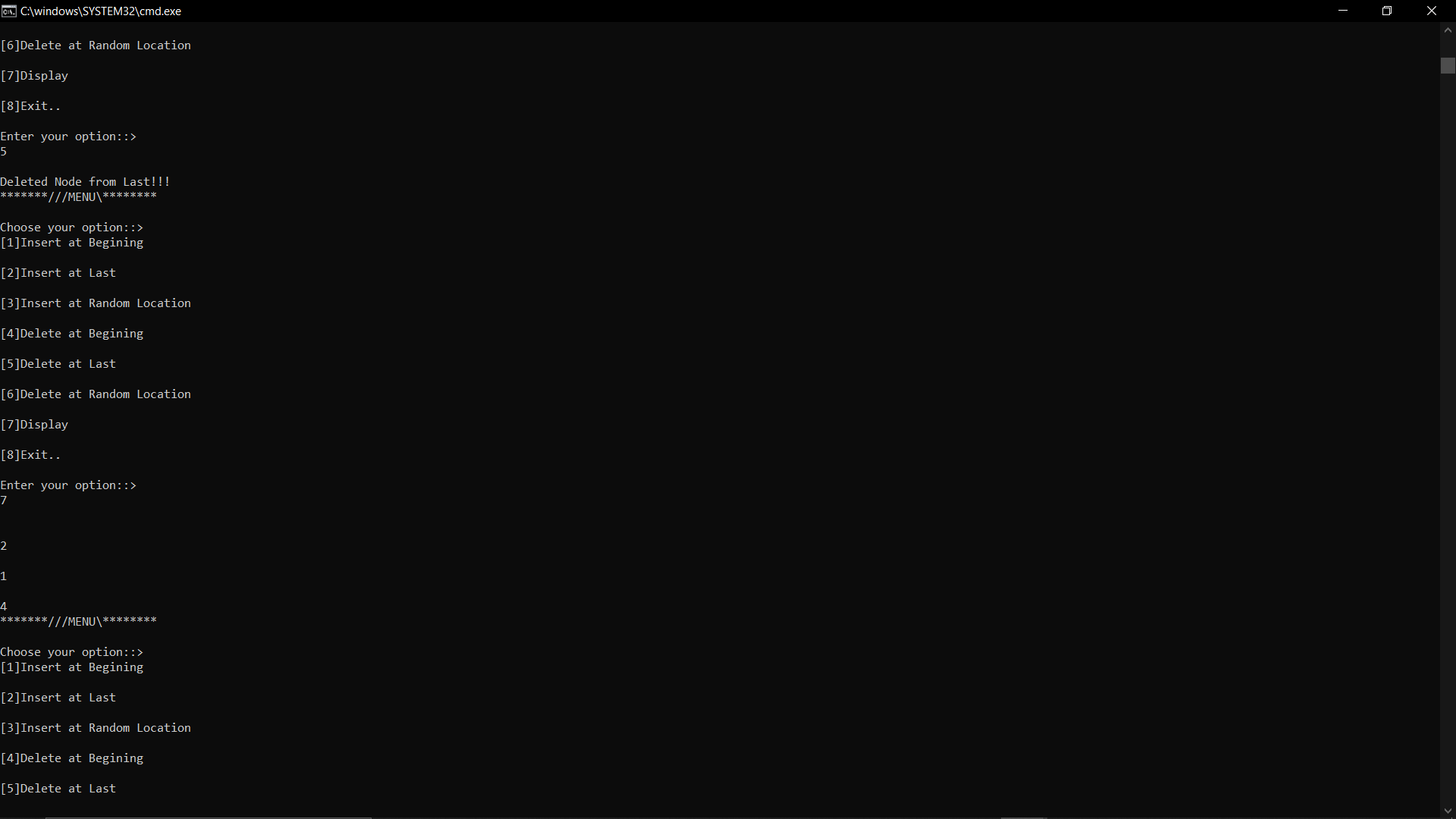
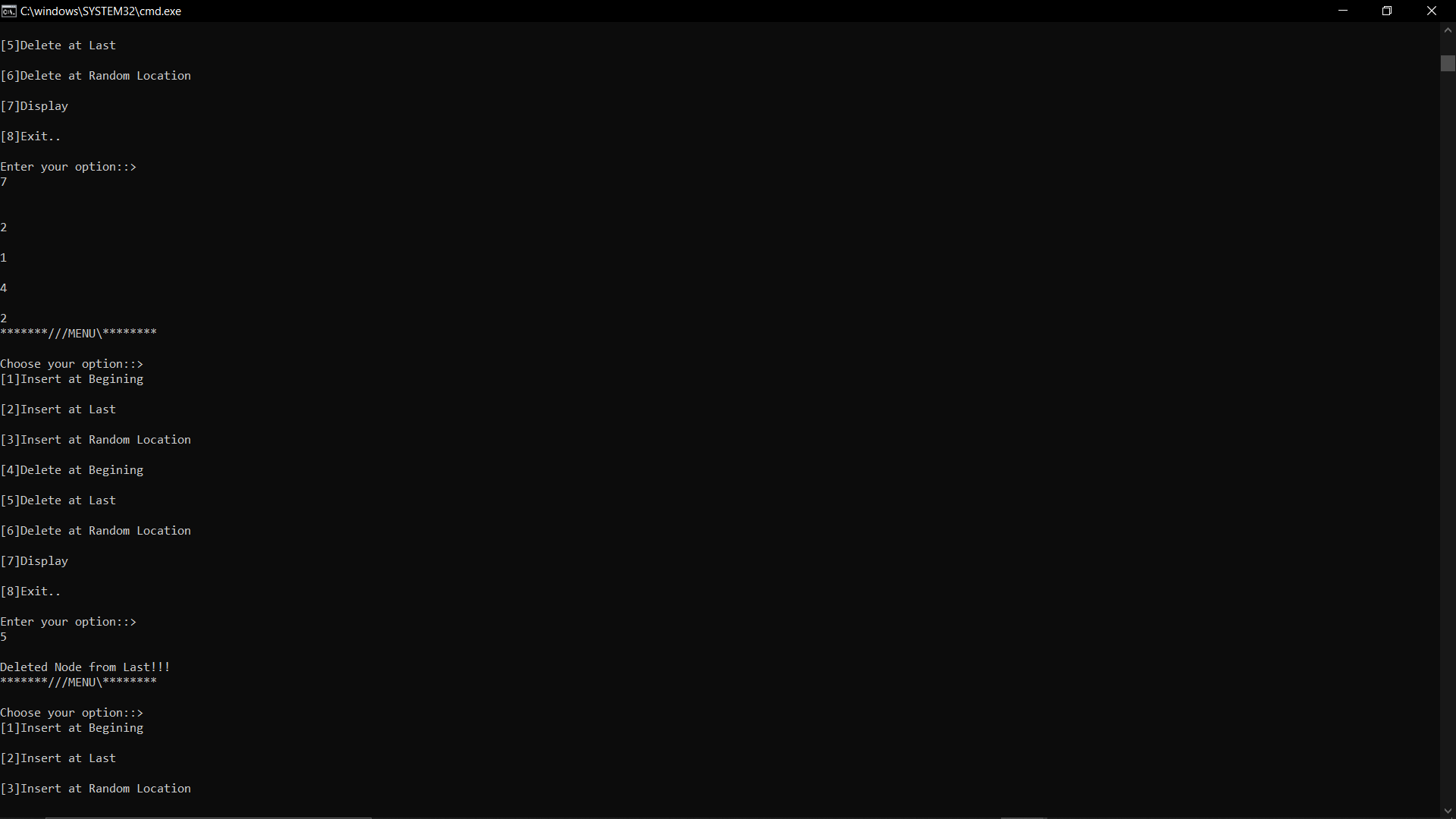
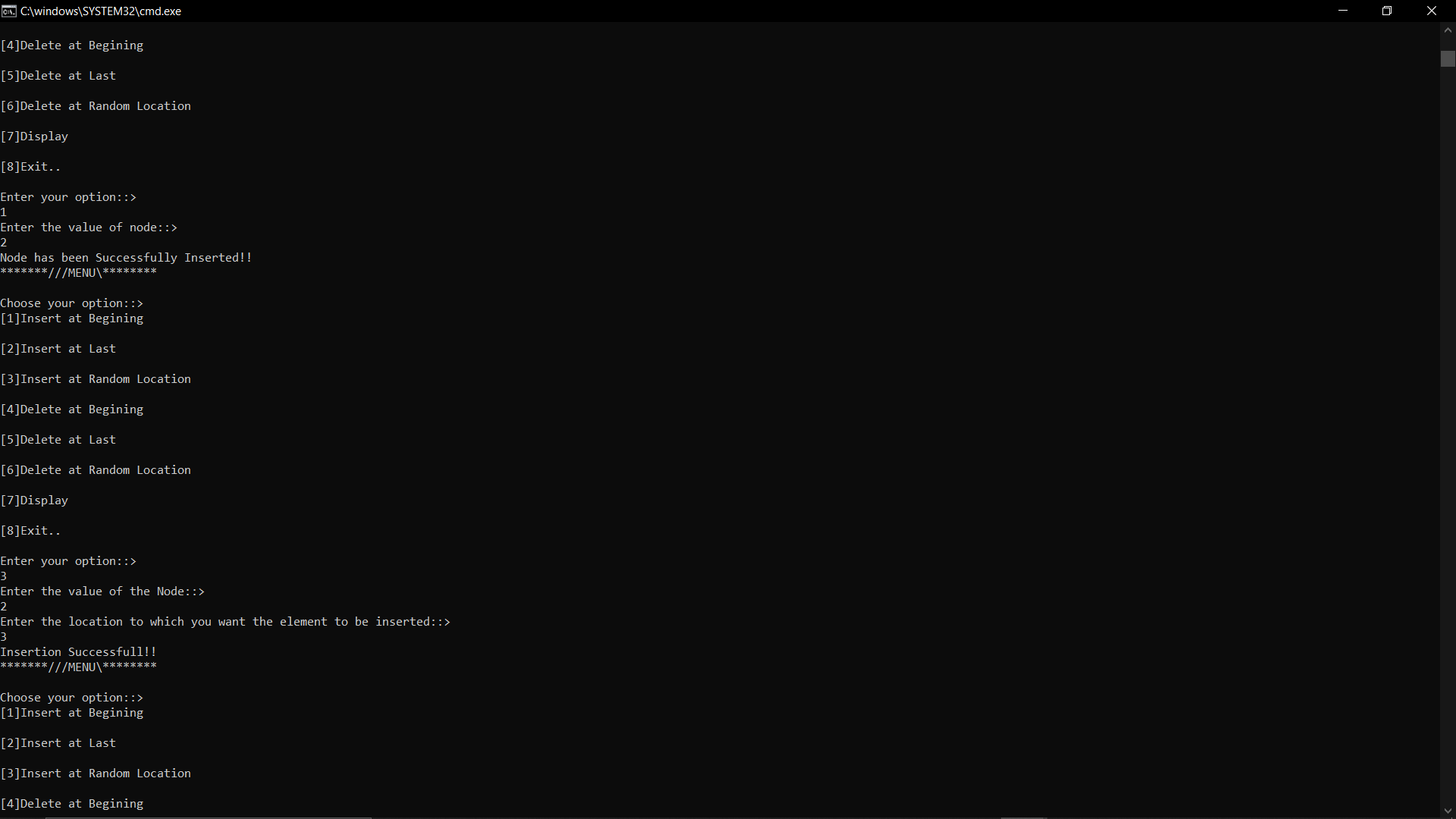
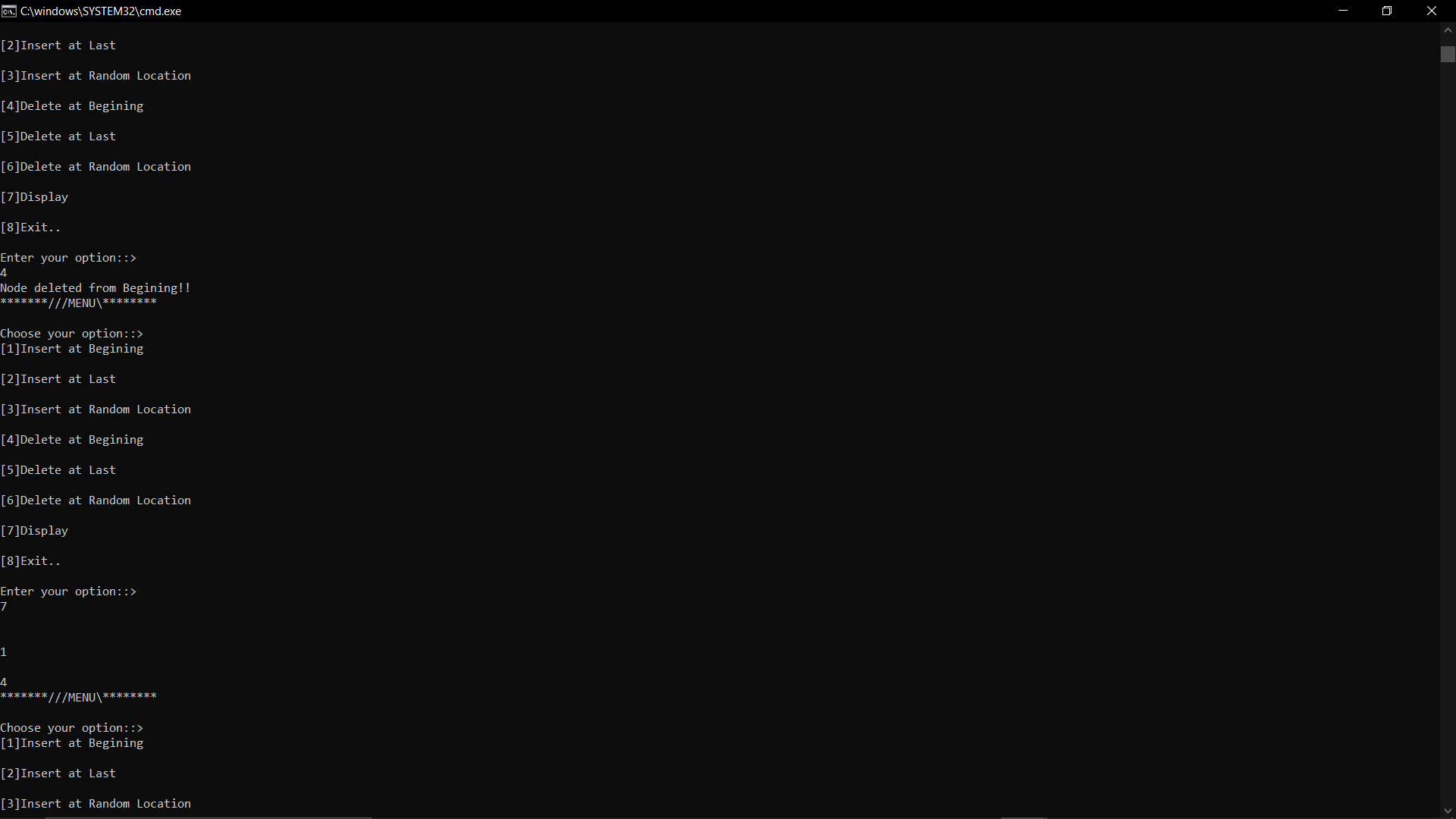
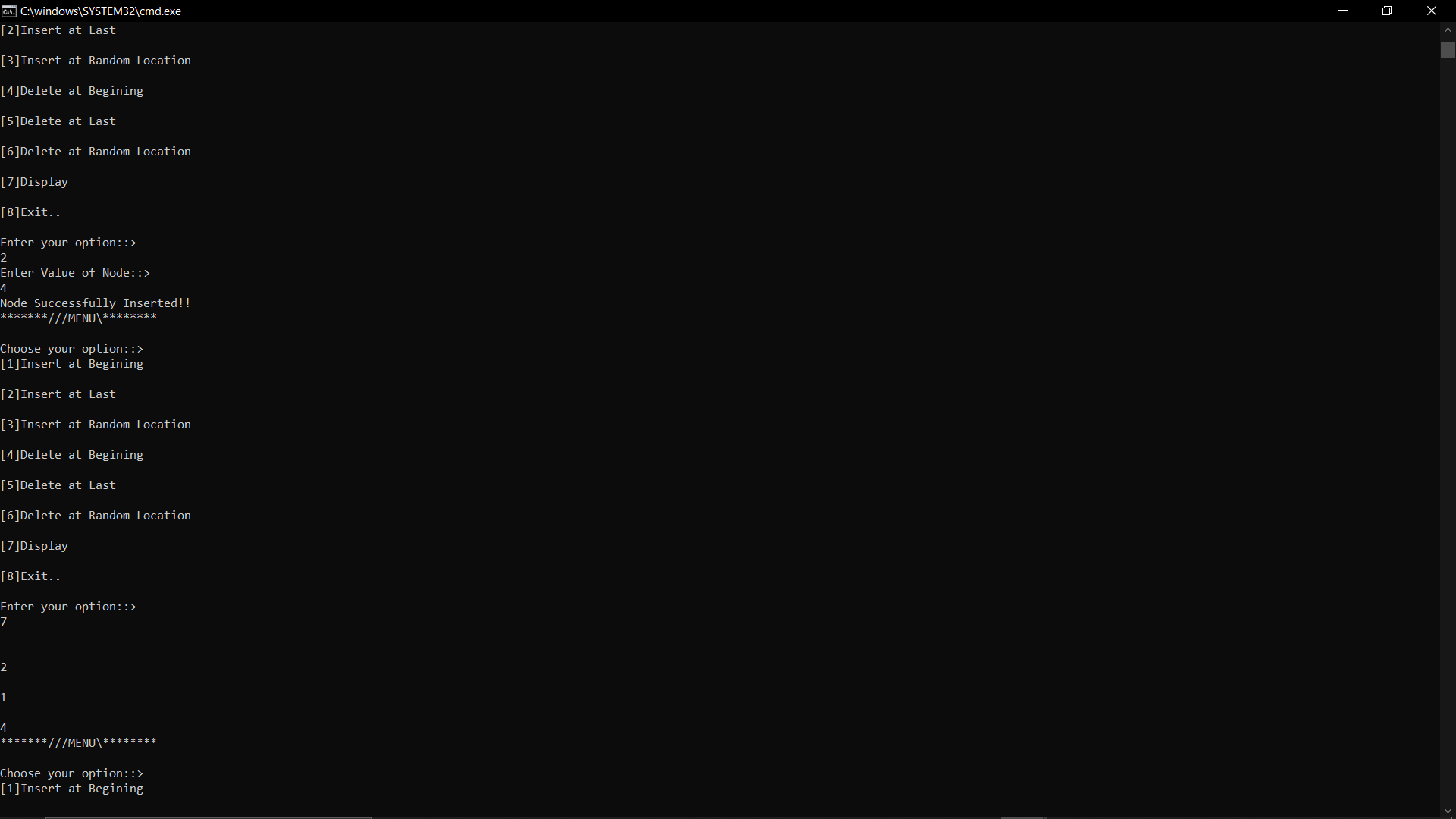
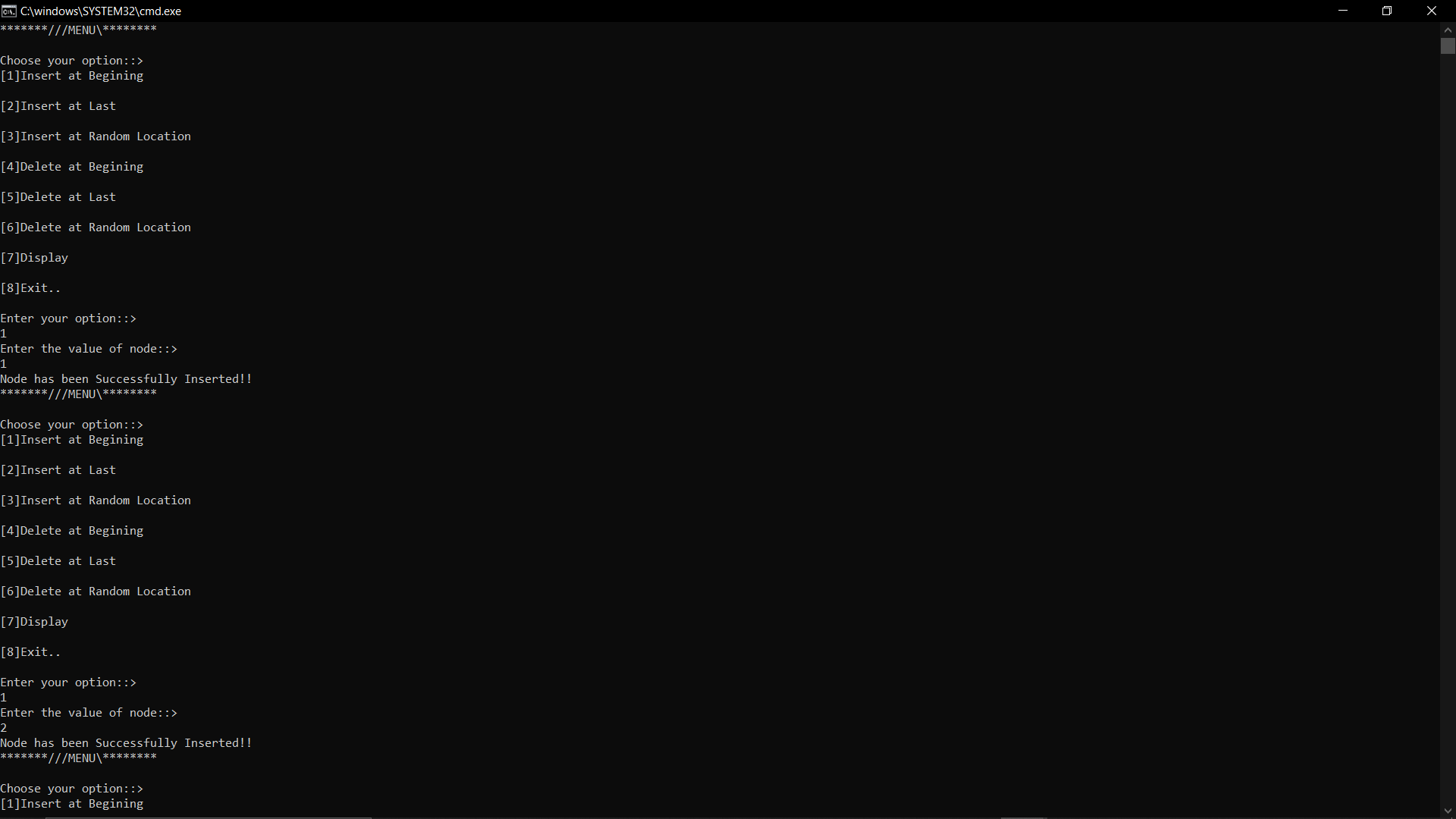
printf("\n%d\n", ptr->data);

ptr = ptr->next;

}

}

}



#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head;

struct node \*head1;

void push()

{

struct node \*ptr;

int new\_data;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW!!!");

}

else

{

printf("\nEnter the Value to be inserted::>>");

scanf("%d",&new\_data);

ptr->data = new\_data;

ptr->next = head;

head = ptr;

printf("\nNode inserted at the top of the Stack\n");

}

}

void pop()

{

struct node \*ptr;

if(head == NULL)

{

printf("LIST IS EMPTY!!!");

}

else

{

ptr = head;

head = ptr->next;

free(ptr);

printf("\nNode deleteed from the top of the Stack\n");

}

}

void enqueue()

{

struct node \*ptr,\*temp;

int new\_data;

ptr = (struct node \*)malloc(sizeof(struct node));

printf("\nEnter the Value to be inserted::>>");

scanf("%d",&new\_data);

ptr->data = new\_data;

if(head == NULL)

{

ptr->next = NULL;

head = ptr;

printf("\nNode inserted at the front of the Queue\n");

}

else

{

temp = head;

while(temp->next != NULL)

{

temp = temp->next;

}

temp->next = ptr;

ptr->next = NULL;

printf("\nNode inserted at the rear of the Queue\n\n");

}

}

void dequeue()

{

struct node \*ptr;

if(head == NULL)

{

printf("LIST IS EMPTY!!!");

}

else

{

ptr = head;

head = ptr->next;

free(ptr);

printf("\nNode deleted from the front of the Queue\n");

}

}

void display()

{

struct node \*ptr;

ptr = head;

if(ptr == NULL)

{

printf("LIST IS EMPTY!!!");

}

else

{

printf("\n\nThe LIST is >>>>>");

while(ptr != NULL)

{

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

ptr = ptr->next;

}

}

}

void sort()

{

struct node \*ptr = head;

struct node \*temp = NULL;

int i;

if(head == NULL)

{

return;

}

else

{

while(ptr != NULL)

{

temp = ptr->next;

while(temp != NULL)

{

if(ptr->data >temp->data)

{

i = ptr->data;

ptr->data = temp->data;

temp->data = i;

}

temp = temp->next;

}

ptr = ptr->next;

}

}

}

void reverse()

{

struct node \*prev = NULL;

struct node \*next = NULL;

struct node \*ptr = head;

while(ptr != NULL)

{

next = ptr->next;

ptr->next = prev;

prev = ptr;

ptr = next;

}

head = prev;

}

struct node \*create\_list(struct node \*head)

{

struct node \*ptr,\*temp;

int i,n,new\_data;

printf("\nEnter the number of nodes ::>> ");

scanf("%d",&n);

head = NULL;

if(n == 0)

{

return head;

}

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

{

ptr = (struct node \*)malloc(sizeof(struct node));

printf("Enter the element to be inserted ::>> ");

scanf("%d",&new\_data);

ptr->data = new\_data;

if(head == NULL)

{

ptr->next = NULL;

head = ptr;

}

else

{

temp = head;

while(temp->next != NULL)

{

temp = temp->next;

}

temp->next = ptr;

ptr->next = NULL;

}

}

return head;

}

struct node \*concatenate(struct node \*head, struct node \*head1)

{

struct node \*ptr;

if(head == NULL)

{

head = head1;

return head;

}

if(head1 == NULL)

{

return head;

}

ptr = head;

while(ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->next = head1;

return head;

}

int main()

{

int option = 0;

while(1)

{

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Choose an option from the list::>>");

printf("\n<-STACK OPERATIONS->\n[1]PUSH\n[2]POP");

printf("\n\n\n<-QUEUE OPERATIONS->\n[3]ENQUEUE\n[4]DEQUEUE");

printf("\n--------------------------");

printf("\n[5]DISPLAY\n[6]SORT\n[7]REVERSE\n[8]CONCATENATION\n[9]EXIT\n");

printf("\nEnter your Option::>>");

scanf("%d",&option);

switch(option)

{

case 1: push();

break;

case 2: pop();

break;

case 3: enqueue();

break;

case 4: dequeue();

break;

case 5: display();

break;

case 6: sort();

printf("\nSorted LIST::>>");

display();

break;

case 7: reverse();

printf("\nReversed LIST::>>");

display();

break;

case 8: printf("\nCreate a Second LIST >>>>>");

head1 = create\_list(head1);

printf("\nLIST 1::>>");

display();

struct node \*ptr;

ptr = head1;

if(ptr == NULL)

{

printf("LIST 2 IS EMPTY!!!");

}

else

{

printf("\n\nLIST 2 >>>>>");

while(ptr != NULL)

{

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

ptr = ptr->next;

}

}

head = concatenate(head,head1);

printf("\n\nConcatenated List::>>");

display();

break;

case 9: exit(1);

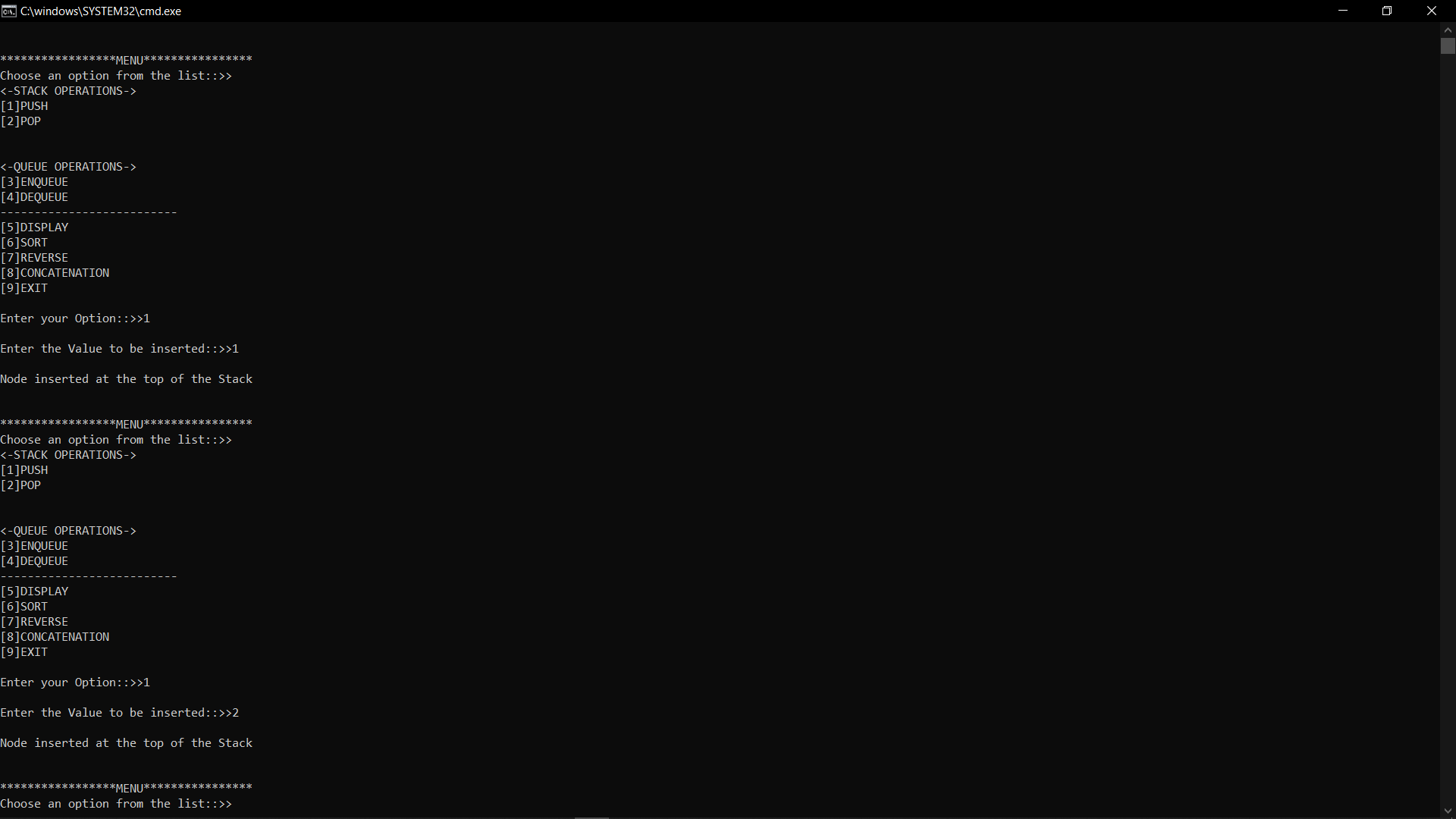
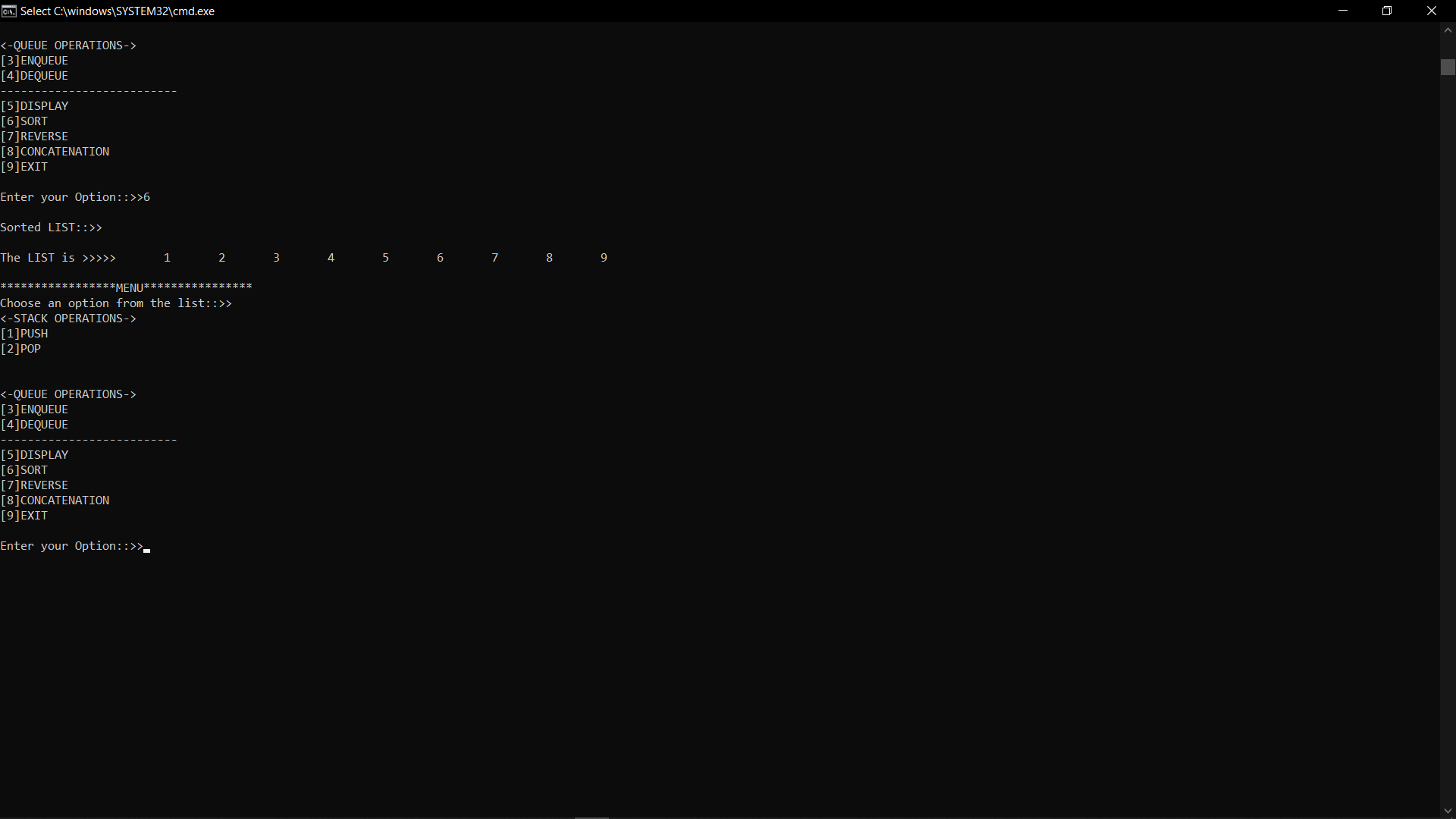
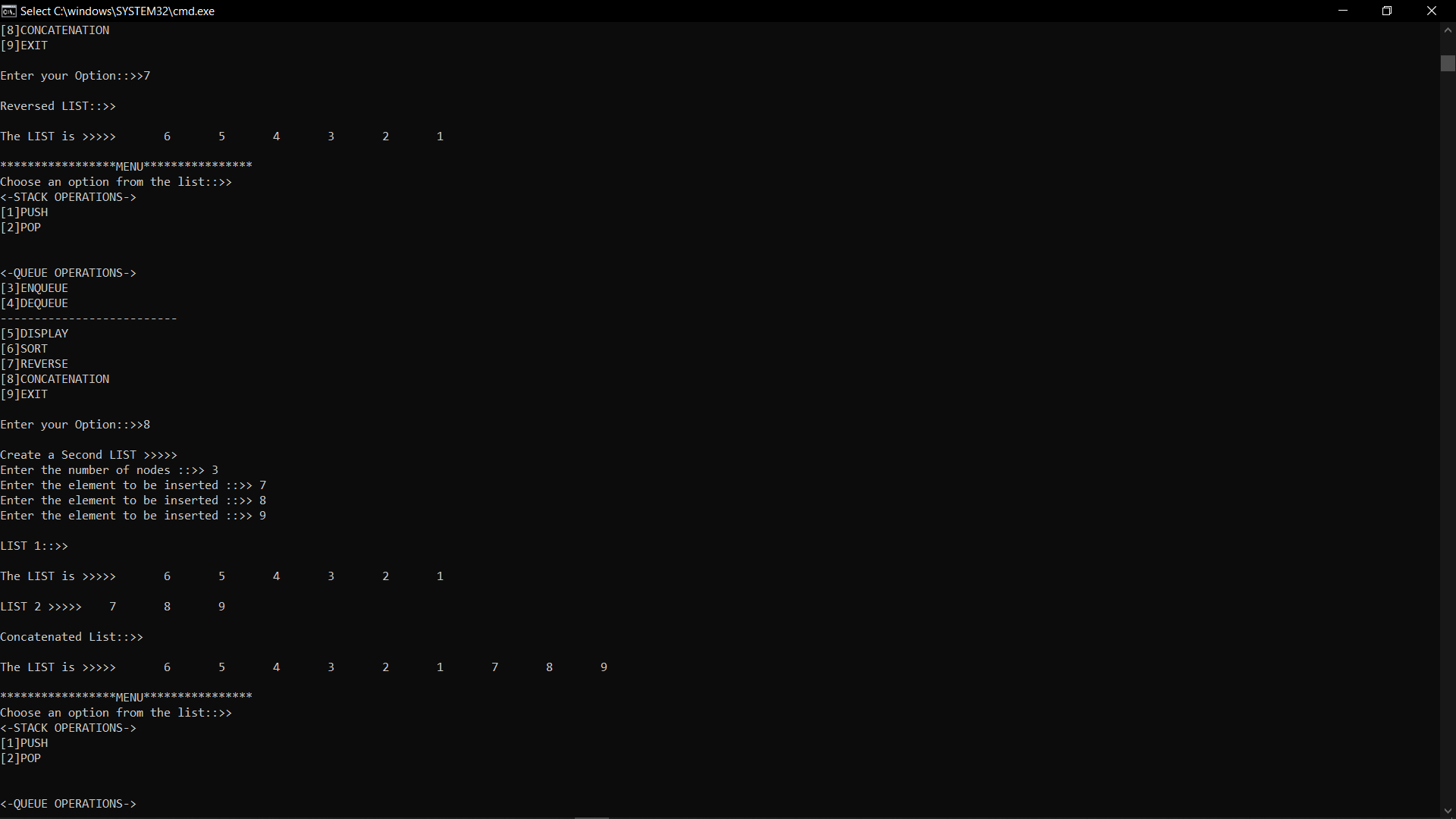
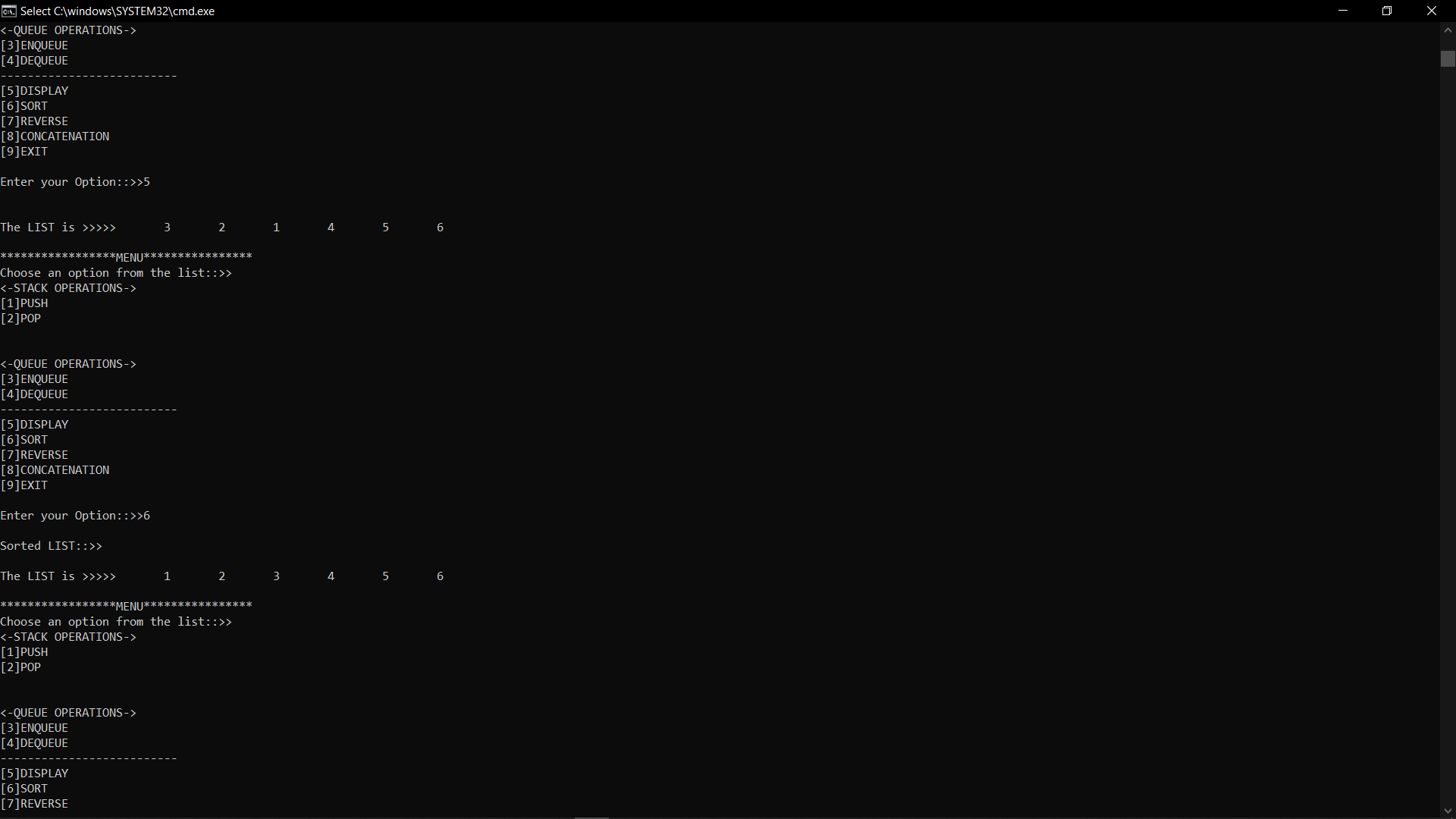
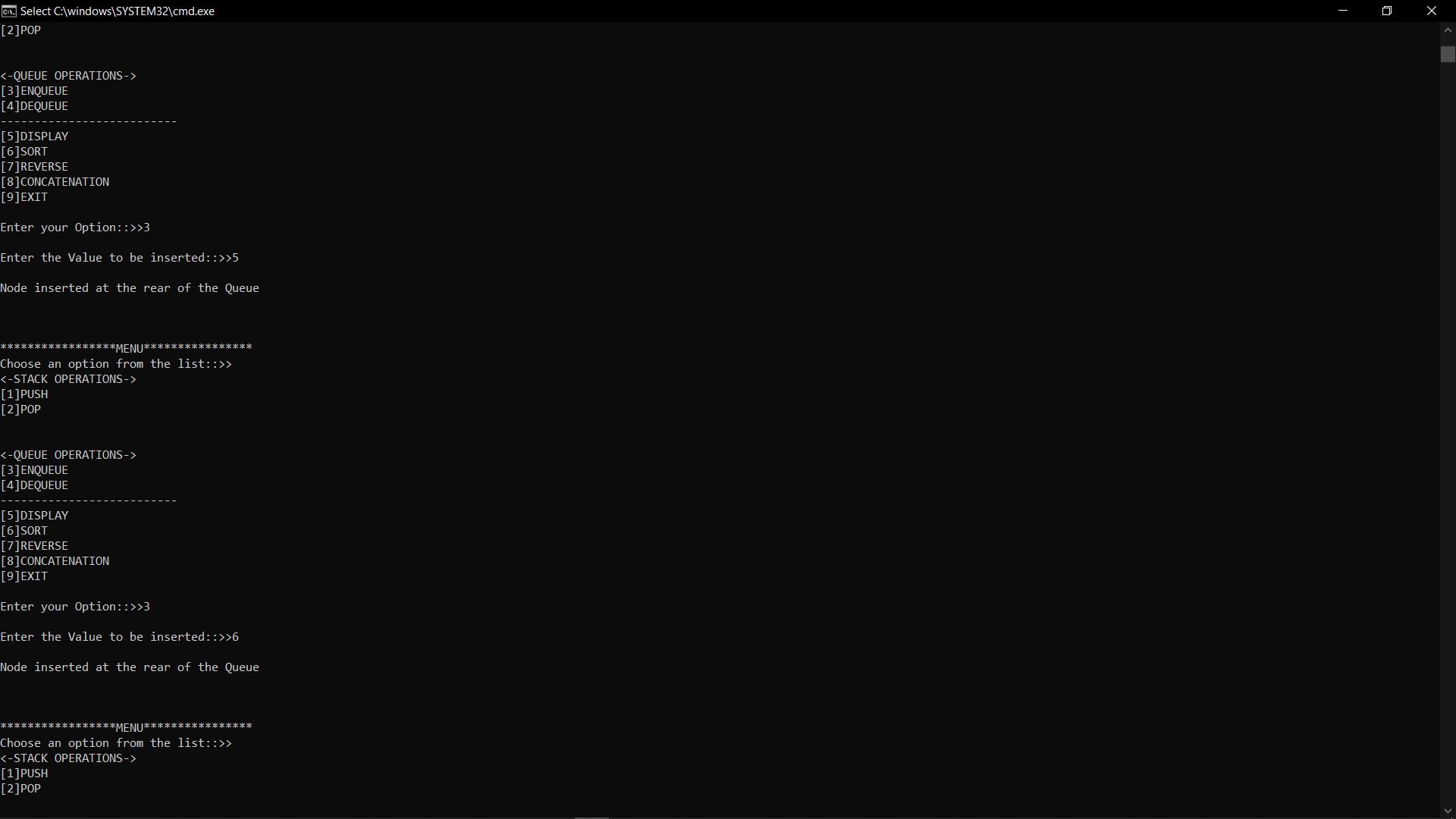
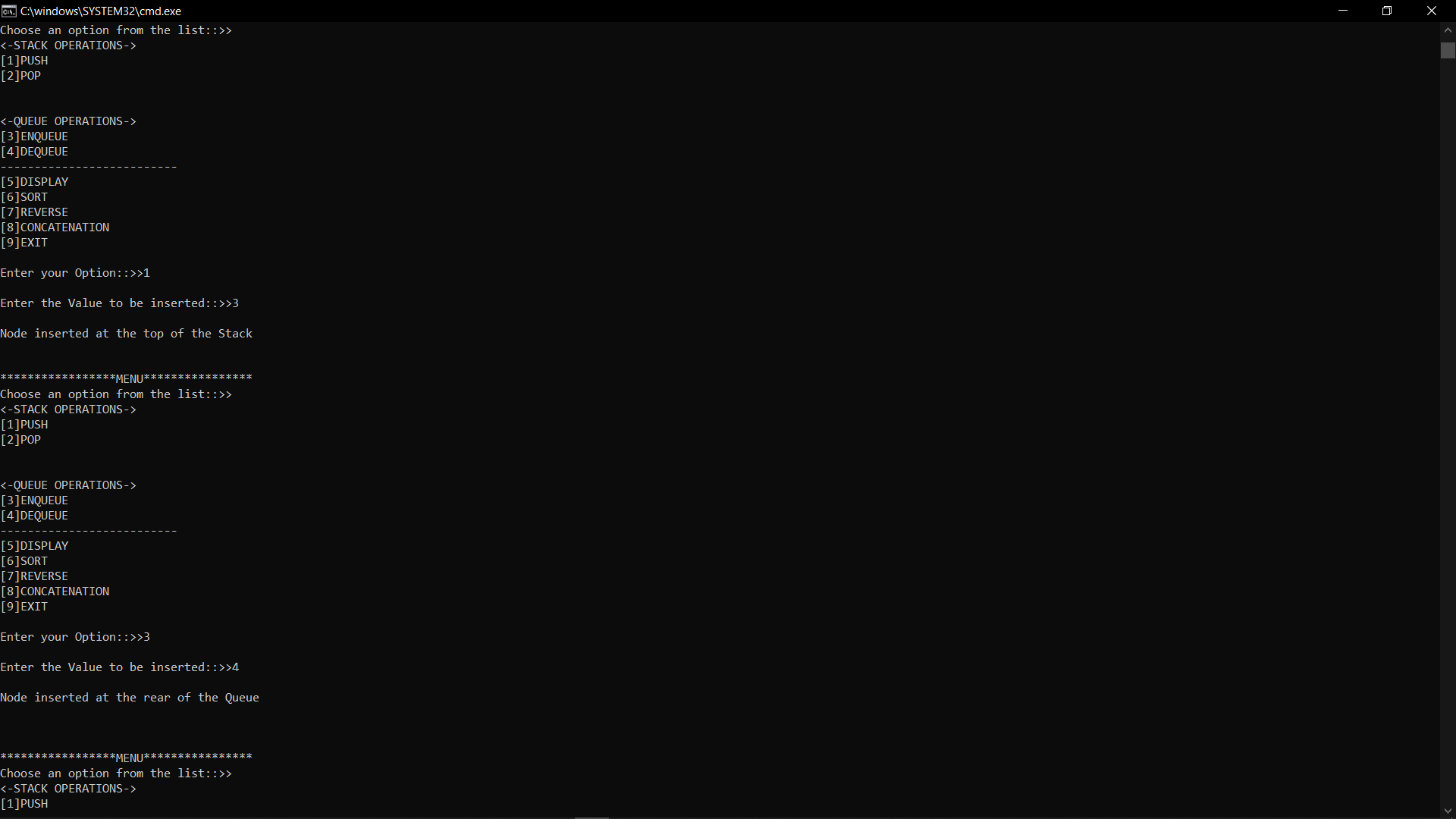
default:

printf("\nINVALID OPTION!!!\n");

}

}

}



#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*head;

void insert\_left(){

int item;

struct node \*ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr==NULL){

printf("LIST IS EMPTY!!!");

ptr->next=NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else{

printf("Enter the value to be added::>\n");

scanf("%d", &item);

printf("Node Inserted at Left!\n");

}

ptr->data=item;

ptr->prev=NULL;

ptr->next=head;

head->prev=ptr;

head=ptr;

}

void delete(){

struct node \*ptr, \*temp;

int value;

printf("Enter the Vakue of positioin to be deleted::>");

scanf("%d", &value);

temp=head;

while(temp->data!=value){

temp=temp->next;

if(temp->next==NULL){

printf("Cannot Delete!!!");

}

else if(temp->next->next==NULL){

temp->next=NULL;

printf("The Node has Been DELETED!!!");

}

else{

ptr=temp->next;

temp->next=ptr->next;

ptr->next->prev=temp;

free(ptr);

printf("The Node has been DELETED!!!");

}

}

}

void display(){

struct node \*ptr;

ptr=head;

if(ptr==NULL){

printf("Nothing to Display\n");

}

else

{

while(ptr!=NULL){

printf("\n%d", ptr->data);

ptr=ptr->next;

}

}

}

int main(){

int option=0;

while(1){

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Choose an option from the list::>>");

printf("\n[1]Insert at Left\n[2]Delete at Specific Position\n[3]Display\n[4]EXIT\n");

printf("\nEnter your Option::>>");

scanf("%d",&option);

switch(option)

{

case 1: insert\_left();

break;

case 2: delete();

break;

case 3: display();

break;

case 4: exit(1);

default:

printf("\nINVALID OPTION!!!\n");

}

}

}

#include<stdlib.h>

#include<stdio.h>

struct node

{

int key;

struct node \*left;

struct node \*right;

};

struct node \*root;

struct node \*create(int data){

struct node \*temp;

temp = (struct node\*)malloc(sizeof(struct node));

temp->key = data;

temp->left = temp->right = NULL;

return temp;

}

void insert(struct node \*root,struct node \*temp){

if(temp->key < root->key){

if(root->left != NULL){

insert(root->left,temp);

}

else{

root->left = temp;

}

}

if(temp->key > root->key){

if(root->right != NULL){

insert(root->right,temp);

}

else{

root->right = temp;

}

}

}

void display(struct node \*root){

if(root != NULL){

display(root->left);

printf("%d\t",root->key);

display(root->right);

}

}

void inorder(struct node \*root){

if(root != NULL){

inorder(root->left);

printf("%d\t",root->key);

inorder(root->right);

}

}

void preorder(struct node \*root){

if(root != NULL){

printf("%d\t",root->key);

preorder(root->left);

preorder(root->right);

}

}

void postorder(struct node \*root){

if(root != NULL){

postorder(root->left);

postorder(root->right);

printf("%d\t",root->key);

}

}

int main(){

char ch;

struct node \*temp;

root = NULL;

int choice = 0;

int data;

while(1){

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Choose an option from the list::>");

printf("\n[1]Create a Tree\n[2]Inorder Trsversal\n[3]Preorder Traversal\n[4]Postrorder Traversal \n[5]Display\n[6]Exit\n");

printf("\nEnter your choice:");

scanf("%d",&choice);

switch(choice){

case 1:

do{

printf("\nEnter the value:");

scanf("%d",&data);

temp = create(data);

if(root == NULL){

root = temp;

}

else{

insert(root,temp);

}

printf("\nDo you Want to Enter more(Y/N)? ");

getchar();

scanf("%c",&ch);

}

while(ch=='y'||ch=='Y');

break;

case 2: printf("\nINORDER TRAVERSAL-->\t");

inorder(root);

break;

case 3: printf("\nPREORDER TRAVERSAL-->\t");

preorder(root);

break;

case 4: printf("\nPOSTORDER TRAVERSAL-->\t");

postorder(root);

break;

case 5: display(root);

break;

case 6: exit(1);

default:

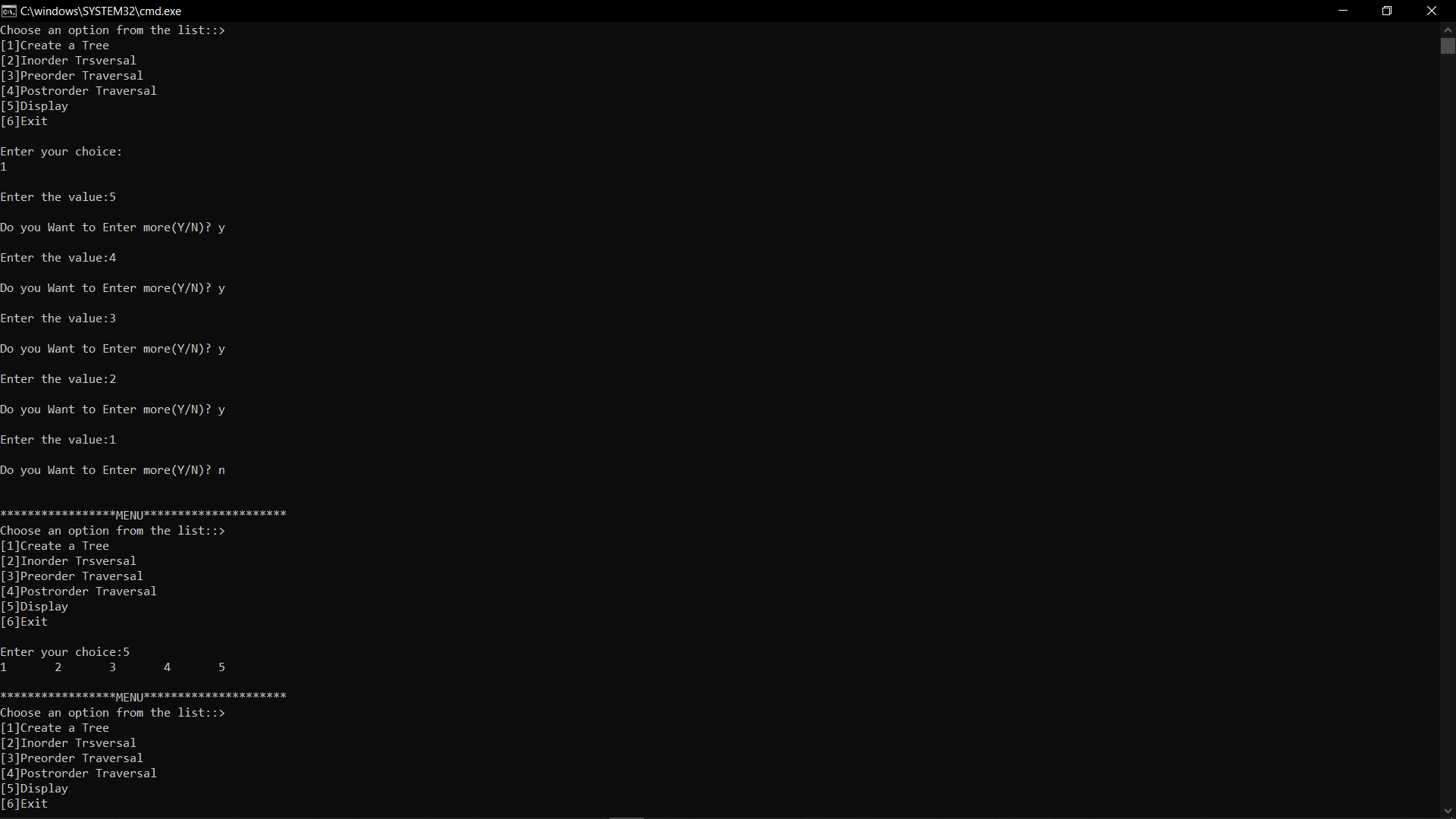
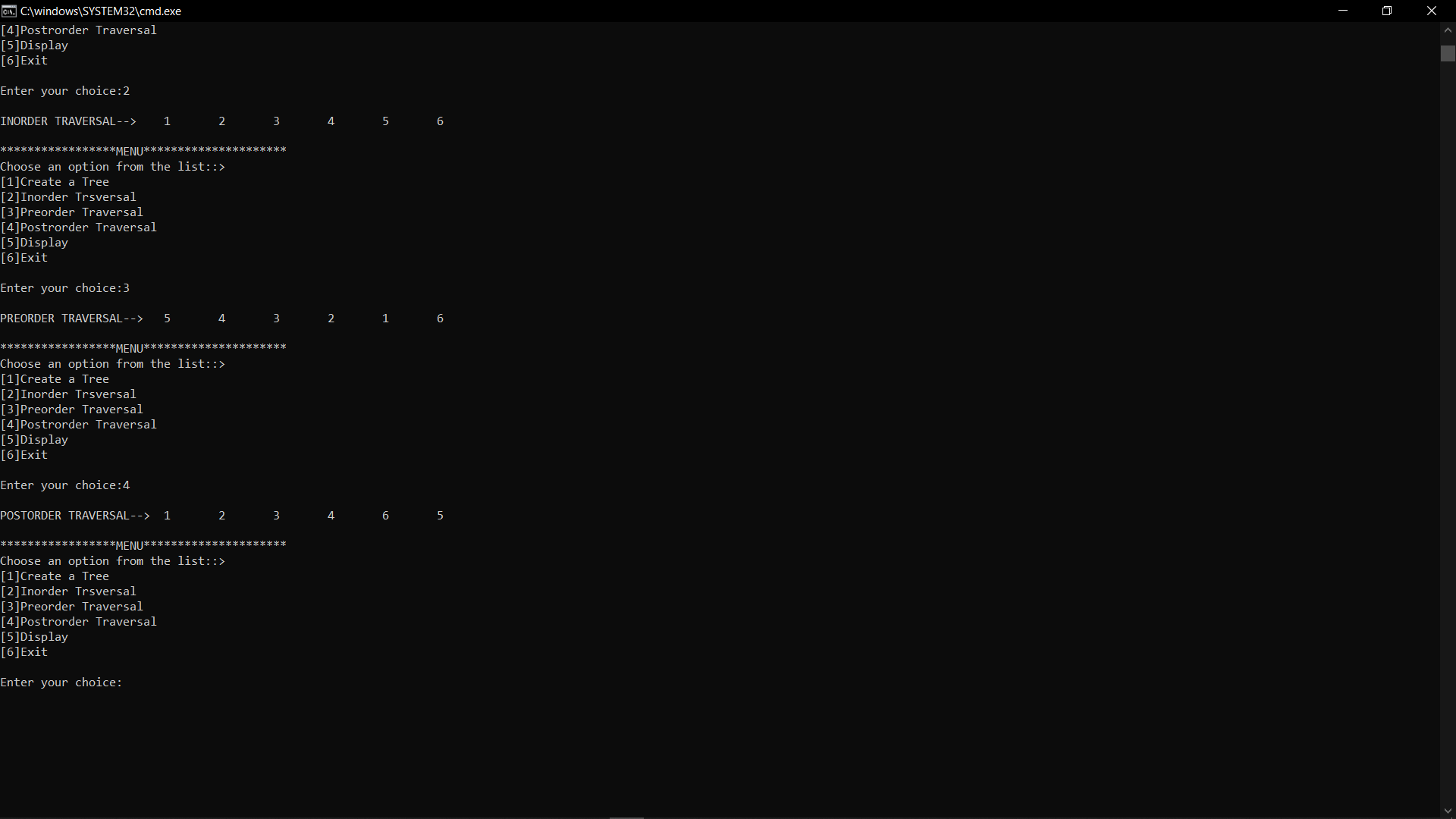
printf("\nINVALID CHOICE!!!\n");

}

}

return 0;

}



#include<stdlib.h>

#include<stdio.h>

struct node

{

int key;

struct node \*left;

struct node \*right;

};

struct node \*root;

struct node \*create(int data)

{

struct node \*temp;

temp = (struct node\*)malloc(sizeof(struct node));

temp->key = data;

temp->left = temp->right = NULL;

return temp;

}

void insert(struct node \*root,struct node \*temp)

{

if(temp->key < root->key)

{

if(root->left != NULL)

{

insert(root->left,temp);

}

else

{

root->left = temp;

}

}

if(temp->key > root->key)

{

if(root->right != NULL)

{

insert(root->right,temp);

}

else

{

root->right = temp;

}

}

}

void display(struct node \*root)

{

if(root != NULL)

{

display(root->left);

printf("%d\t",root->key);

display(root->right);

}

}

void inorder(struct node \*root)

{

if(root != NULL)

{

inorder(root->left);

printf("%d\t",root->key);

inorder(root->right);

}

}

void preorder(struct node \*root)

{

if(root != NULL)

{

printf("%d\t",root->key);

preorder(root->left);

preorder(root->right);

}

}

void postorder(struct node \*root)

{

if(root != NULL)

{

postorder(root->left);

postorder(root->right);

printf("%d\t",root->key);

}

}

int main()

{

char ch;

struct node \*temp;

root = NULL;

int choice = 0;

int data;

while(1)

{

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Choose an option from the list:");

printf("\n[1]CREATE A TREE\n[2]INORDER TRAVERSAL\n[3]PREORDER TRAVERSAL\n[4]POSTORDER TRAVERSAL\n[5]DISPLAY\n[6]EXIT\n");

printf("\nEnter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1: do{

printf("\nEnter the value:");

scanf("%d",&data);

temp = create(data);

if(root == NULL)

{

root = temp;

}

else

{

insert(root,temp);

}

printf("\nDo you Want to Enter more(Y/N)? ");

getchar();

scanf("%c",&ch);

}while(ch=='y'||ch=='Y');

break;

case 2: printf("\nINORDER TRAVERSAL-->\t");

inorder(root);

break;

case 3: printf("\nPREORDER TRAVERSAL-->\t");

preorder(root);

break;

case 4: printf("\nPOSTORDER TRAVERSAL-->\t");

postorder(root);

break;

case 5: display(root);

break;

case 6: exit(1);

default:

printf("\nINVALID CHOICE!!!\n");

}

}

return 0;

}

Text

Description automatically generatedText

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence