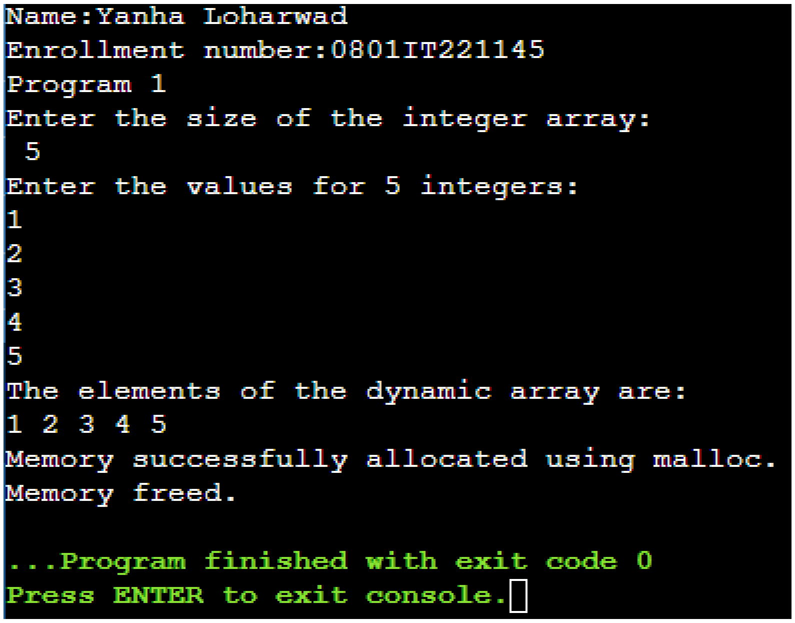
# Program 1

1. **Write a program to create dynamic int array using malloc() and free().**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int size;
printf("Name:Yanha Loharwad\n");
printf("Enrollment number:0801IT221145\n");
printf("Program 1\n");
printf("Enter the size of the integer array:\n ");
scanf("%d", &size);
int *da = (int *)malloc(size * sizeof(int));

if (da == 0)
{
   printf("Memory allocation failed.\n");
   return 1;
}
printf("Enter the values for %d integers:\n", size);
for (int i = 0; i < size; i++)
{
   scanf("%d", &da[i]);
}
printf("The elements of the dynamic array are:\n");
```

```
    for (int i = 0; i < size; i++)

    {

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

    }

    printf("\n");

    printf("Memory successfully allocated using malloc.\n");

    free(da);

    printf("Memory freed.");

    return 0;

}
```

**OUTPUT: Memory successfully allocated using malloc() and freed using free().**



```
Name:Yanha Loharwad
Enrollment number:0801IT221145
Program 1
Enter the size of the integer array:
 5
Enter the values for 5 integers:
1
2
3
4
5
The elements of the dynamic array are:
1 2 3 4 5
Memory successfully allocated using malloc.
Memory freed.

...Program finished with exit code 0
Press ENTER to exit console.
```

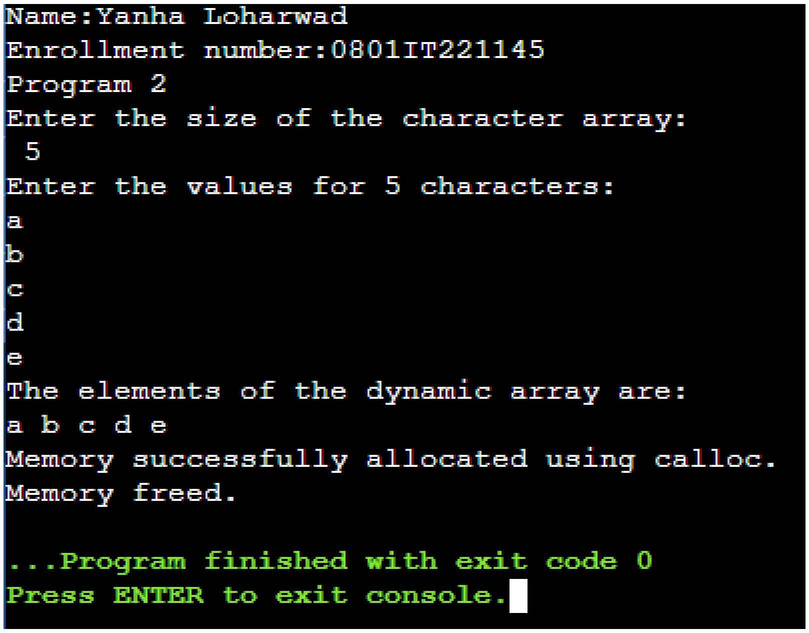
# Program 2

**2. Write a program to create dynamic char array using calloc() and free().**

```
     #include <stdio.h>
    #include <stdlib.h>
  int main()
    {
  int size;
  printf("Name:Yanha Loharwad\n");
  printf("Enrollment number:0801IT221145\n");
  printf("Program 2\n");
  printf("Enter the size of the character array:\n ");
  scanf("%d", &size);
char *da = (char *)calloc(size,sizeof(char));
if (da == 0)
{
    printf("Memory allocation failed.\n");
    return 1;
}
printf("Enter the values for %d characters:\n", size);
for (int i = 0; i < size; i++)
{
    scanf(" %c", &da[i]);
}
```

```
    printf("The elements of the dynamic array are:\n");
    for (int i = 0; i < size; i++)
    {
        printf("%c ", da[i]);
    }
    printf("\n");
    printf("Memory successfully allocated using calloc.\n");
    free(da);
    printf("Memory freed.");
    return 0;
}
```

**OUTPUT: Memory successfully allocated using calloc() and freed using free().**



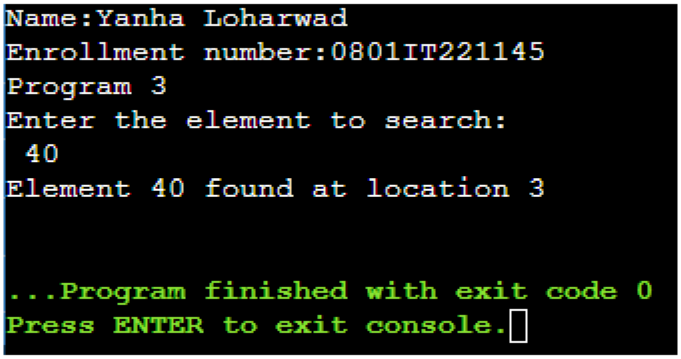
```
Name:Yanha Loharwad
Enrollment number:0801IT221145
Program 2
Enter the size of the character array:
 5
Enter the values for 5 characters:
a
b
c
d
e
The elements of the dynamic array are:
a b c d e
Memory successfully allocated using calloc.
Memory freed.

...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 3

3. **Write a program to implement linear search.**

```
    #include <stdio.h>

    #include <stdlib.h>

  int main()

    {

  int arr[] ={10,20,30,40,50,60,70,80};

  int key,i;

  printf("Name:Yanha Loharwad\n");

  printf("Enrollment number:0801IT221145\n");

  printf("Program 3\n");

  printf("Enter the element to search:\n ");

  scanf("%d", &key);

  int found=0;

 for (i = 0; i < sizeof(arr)/sizeof(int); i++)

 {

   if(arr[i]==key)

   {

   found=1;

   break;

 }

 }

if(found==1)

{
    printf("Element %d found at location %d \n",key,i);
  }
  else
  {
    printf("Element %d not found.",key);
  }
   return 0;
}
```

**OUTPUT: Element is found using linear search.**



```
Name:Yanha Loharwad
Enrollment number:0801IT221145
Program 3
Enter the element to search:
 40
Element 40 found at location 3


...Program finished with exit code 0
Press ENTER to exit console.
```

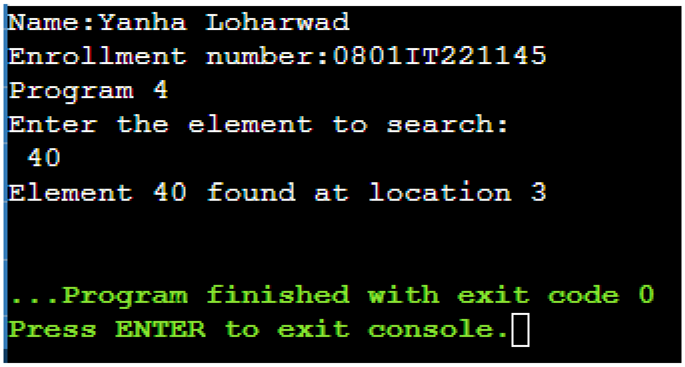
# Program 4

**4. Write a program to implement binary search.**

```
    #include <stdio.h>
   #include <stdlib.h>
 int main()
   {
 int arr[] ={10,20,30,40,50,60,70,80,90,100};
 int size=sizeof(arr)/sizeof(int);
 int key,i,found=0;
 int low=0;
 int high=size-1;
 printf("Name:Yanha Loharwad\n");
 printf("Enrollment number:0801IT221145\n");
 printf("Program 4\n");
 printf("Enter the element to search:\n ");
 scanf("%d", &key);
while(low<=high)
{
   int mid=low + (high-low)/2;
    if (arr[mid] == key) {
      found = 1;
      i = mid;
      break;
   }
```

```
        if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    if (found) {
        printf("Element %d found at location %d\n", key, i);
    } else {
        printf("Element %d not found in the array\n", key);
    }
    return 0;
}
```

**OUTPUT: Element is found using binary search.**



```
Name:Yanha Loharwad
Enrollment number:0801IT221145
Program 4
Enter the element to search:
 40
Element 40 found at location 3


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 5

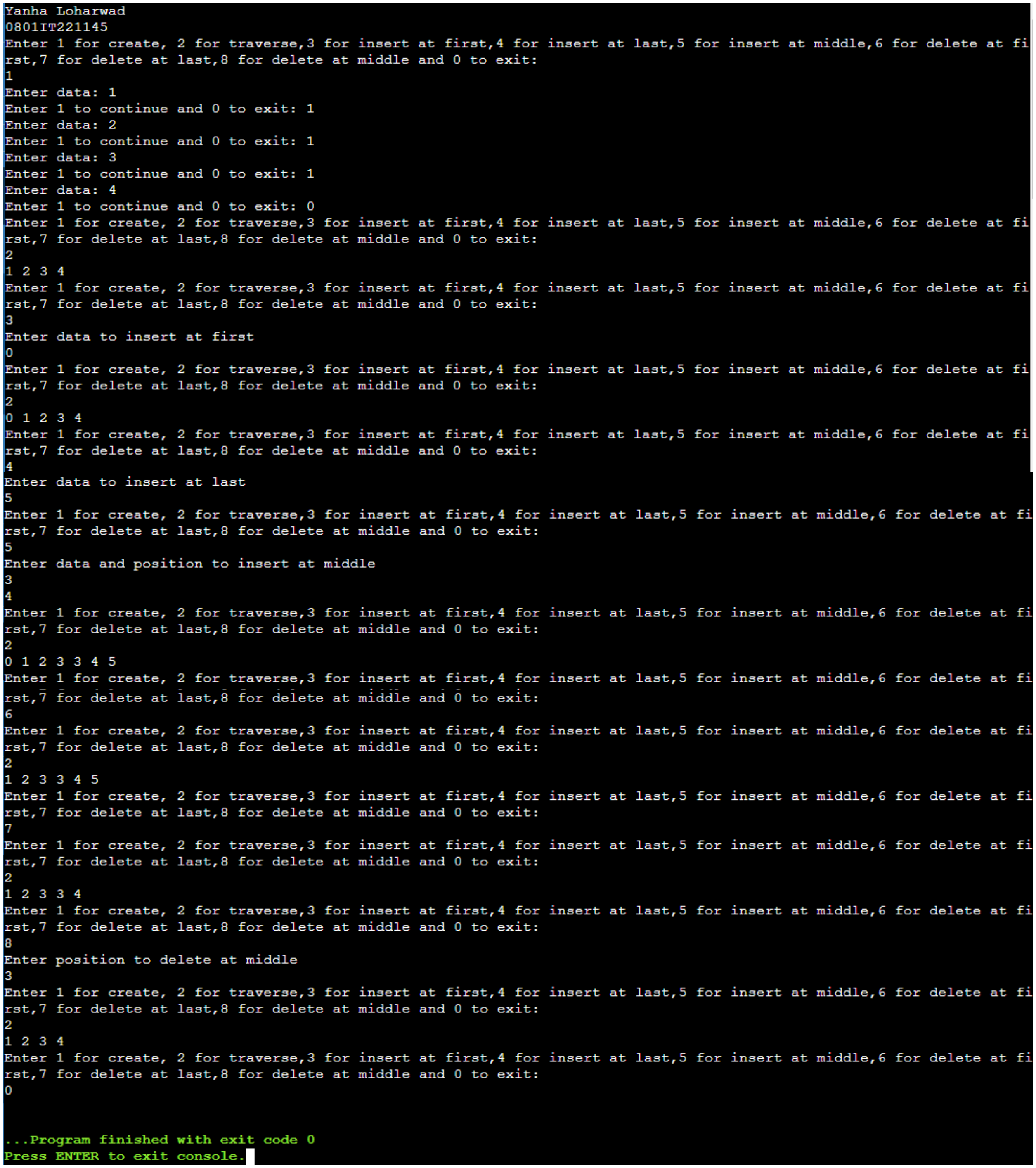
5. **Creation, traversing, insertion at first, insertion at last ,insertion at any position, deletion at first, deletion at last, deletion at any position of a singly linked list.**

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
int data;
struct node *next;
};

struct node *head, *newnode, *temp;

void create()
{
head = NULL;
int choice;
do
{
newnode = (struct node *)malloc(sizeof(struct node));
printf("Enter data: ");
scanf("%d", &newnode->data);
newnode->next = NULL;
if (head == NULL)
{
head = temp = newnode;
}
else
{
temp->next = newnode;
temp = newnode;
}
printf("Enter 1 to continue and 0 to exit: ");
scanf("%d", &choice);
} while (choice != 0);
}

void traverse()
```

```
{
temp = head;
while (temp != NULL)
{
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
void insertAtFirst(int data)
{
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->next=head;
head=newnode;
}
void insertAtLast(int data)
{
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
newnode->next=NULL;
temp=head;
while(temp->next!=NULL)
{
temp=temp->next;
}
temp->next=newnode;
}
void insertAtMiddle(int data,int position)
{
newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=data;
temp=head;
int count=1;
while(count<position-1)
{
temp=temp->next;
count++;
}
newnode->next=temp->next;
temp->next=newnode;
}
void deleteAtFirst()
{
temp=head;
head=head->next;
free(temp);
}
void deleteAtLast()
{
struct node *prevnode;
temp=head;
while(temp->next!=NULL)
{
prevnode=temp;
temp=temp->next;
}
if(temp==head)
{
head=NULL;
}
else{
prevnode->next=NULL;
}
free(temp);
}
void deleteAtMiddle(int position)
{
struct node *nextnode;
int count = 1;
temp = head;
while (count < position - 1)
{
temp = temp->next;
count++;
}
nextnode = temp->next;
temp->next = nextnode->next;
free(nextnode);
}
```

```
void main()
{   printf("Yanha Loharwad\n0801IT221145\n");
int choice;
head = NULL;
while (1)
{
printf("Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at first,7 for delete at last,8 for delete at middle and 0 to exit: \n");
scanf("%d", &choice);

switch (choice)
{
case 1:
create();
break;
case 2:
if (head == NULL)
{
printf("Linked list is empty. \n");
}
else
{
traverse();
}
break;
case 3:
{
int data;
printf("Enter data to insert at first\n");
scanf("%d",&data);
insertAtFirst(data);
}
break;
case 4:
{
int data;
printf("Enter data to insert at last\n");
scanf("%d",&data);
insertAtLast(data);
}
break;
case 5:
{
int data,position;
printf("Enter data and position to insert at middle\n");
scanf("%d%d",&data,&position);
insertAtMiddle(data,position);
}
break;
case 6:
deleteAtFirst();
break;
case 7:
deleteAtLast();
break;
case 8:
{
int position;
printf("Enter position to delete at middle\n");
scanf("%d",&position);
deleteAtMiddle(position);
}
break;
case 0:
exit(0);
default:
printf("Enter a valid choice\n");
}
}
}
```

**OUTPUT: Singly linked list successfully created, traversed, node inserted at first, last and any position and node deleted at first, last and at any position.**



```
Yanha Loharwad
0801IT221145
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
1
Enter data: 1
Enter 1 to continue and 0 to exit: 1
Enter data: 2
Enter 1 to continue and 0 to exit: 1
Enter data: 3
Enter 1 to continue and 0 to exit: 1
Enter data: 4
Enter 1 to continue and 0 to exit: 0
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
3
Enter data to insert at first
0
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
2
0 1 2 3 4
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
4
Enter data to insert at last
5
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
5
Enter data and position to insert at middle
3
4
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
2
0 1 2 3 3 4 5
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
6
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
2
1 2 3 3 4 5
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
7
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
2
1 2 3 3 4
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
8
Enter position to delete at middle
3
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse,3 for insert at first,4 for insert at last,5 for insert at middle,6 for delete at fi
rst,7 for delete at last,8 for delete at middle and 0 to exit:
0


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 6

**6. Creation, traversing, insertion at first, insertion at last ,insertion at any position, deletion at first, deletion at last, deletion at any position and reversing of a doubly linked list.**

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node *next;
struct node *prev;
};
struct node *head, *tail, *newnode, *temp;
void create(){
head = NULL;
tail = NULL;
int choice;
do{
newnode = (struct node *)malloc(sizeof(struct node));
printf("Enter data: ");
scanf("%d", &newnode->data);
newnode->next = NULL;
newnode->prev = tail;
if (tail == NULL)
{
head = tail = newnode;
}
else
{
tail->next = newnode;
tail = newnode;
}
printf("Enter 1 to continue and 0 to exit: ");
scanf("%d", &choice);
} while (choice != 0);
}
void traverse(){
temp = head;
while (temp != NULL)
{
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
void insertAtFirst(int data){
newnode = (struct node *)malloc(sizeof(struct node));
newnode->data = data;
newnode->next = head;
newnode->prev = NULL;
if (head != NULL)
```

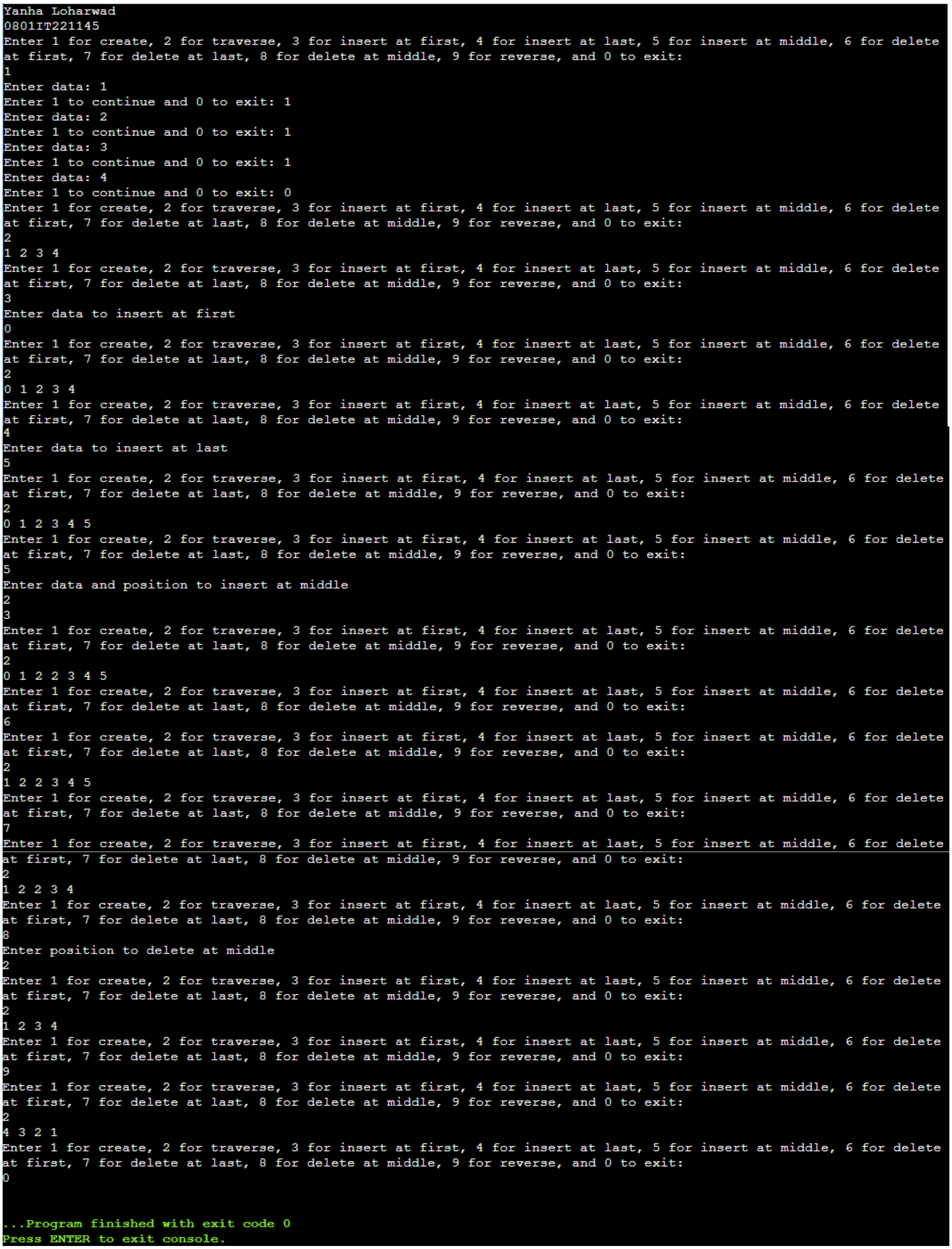
```
{
head->prev = newnode;
}
head = newnode;
}
void insertAtLast(int data){
newnode = (struct node *)malloc(sizeof(struct node));
newnode->data = data;
newnode->next = NULL;
newnode->prev = tail;
if (tail != NULL)
{
tail->next = newnode;
}
tail = newnode;
}
void insertAtMiddle(int data, int position){
newnode = (struct node *)malloc(sizeof(struct node));
newnode->data = data;
temp = head;
int count = 1;
while (count < position - 1 && temp != NULL)
{
temp = temp->next;
count++;
}
newnode->next = temp->next;
newnode->prev = temp;
temp->next = newnode;
if (newnode->next != NULL)
{
newnode->next->prev = newnode;
}
}
void deleteAtFirst(){
if (head == NULL)
{
printf("Linked list is empty. \n");
return;
}
temp = head;
head = head->next;
if (head != NULL)
{
head->prev = NULL;
}
free(temp);
}
void deleteAtLast()
{
```

```
if (tail == NULL)
{
printf("Linked list is empty. \n");
return;
}
temp = tail;
tail = tail->prev;
if (tail != NULL)
{
tail->next = NULL;
}
free(temp);
}
void deleteAtMiddle(int position){
if (head == NULL)
{
printf("Linked list is empty. \n");
}
temp = head;
int count = 1;
while (count < position - 1 && temp != NULL)
{
temp = temp->next;
count++;
}
```

```
struct node *nextnode = temp->next;
temp->next = nextnode->next;
if (temp->next != NULL)
{
temp->next->prev = temp;
}
free(nextnode);
}
void reverse(){
struct node *current = head;
struct node *tempNode = NULL;
while (current != NULL)
{
tempNode = current->prev;
current->prev = current->next;
current->next = tempNode;
current = current->prev;
}
head = tail;
tail = current;
}
void main(){
 printf("Yanha Loharwad\n0801IT221145\n");
int choice;
head = NULL;
```

```
tail = NULL;
while (1)
{
printf("Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit: \n");
scanf("%d", &choice);
switch (choice)
{
case 1:create();break;
case 2:
if (head == NULL){
printf("Linked list is empty. \n");
}
else{
traverse();
}break;
case 3:{
int data;
printf("Enter data to insert at first\n");
scanf("%d", &data);
insertAtFirst(data);
}break;
case 4:{
int data;
printf("Enter data to insert at last\n");
scanf("%d", &data);
insertAtLast(data);
}break;
case 5:{
int data, position;
printf("Enter data and position to insert at middle\n");
scanf("%d%d", &data, &position);
insertAtMiddle(data, position);
}break;
case 6:deleteAtFirst();break;
case 7:deleteAtLast();break;
case 8:{
int position;
printf("Enter position to delete at middle\n");
scanf("%d", &position);
deleteAtMiddle(position);
}break;
case 9:reverse();break;
case 0:exit(0);
default:printf("Enter a valid choice\n");
}
}
}
```

**OUTPUT: Doubly linked list successfully created, traversed, node inserted at first, last and any position, node deleted at first, last and at any position and reversed.**



```
Yanha Loharwad
0801IT221145
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
1
Enter data: 1
Enter 1 to continue and 0 to exit: 1
Enter data: 2
Enter 1 to continue and 0 to exit: 1
Enter data: 3
Enter 1 to continue and 0 to exit: 1
Enter data: 4
Enter 1 to continue and 0 to exit: 0
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
3
Enter data to insert at first
0
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
4
Enter data to insert at last
5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
5
Enter data and position to insert at middle
2
3
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
6
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
7
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
8
Enter position to delete at middle
2
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
9
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
4 3 2 1
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
0


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 7

7. **Creation, traversing, insertion at first, insertion at last ,insertion at any position, deletion at first, deletion at last, deletion at any position,reversing of a singly circular linked list.**

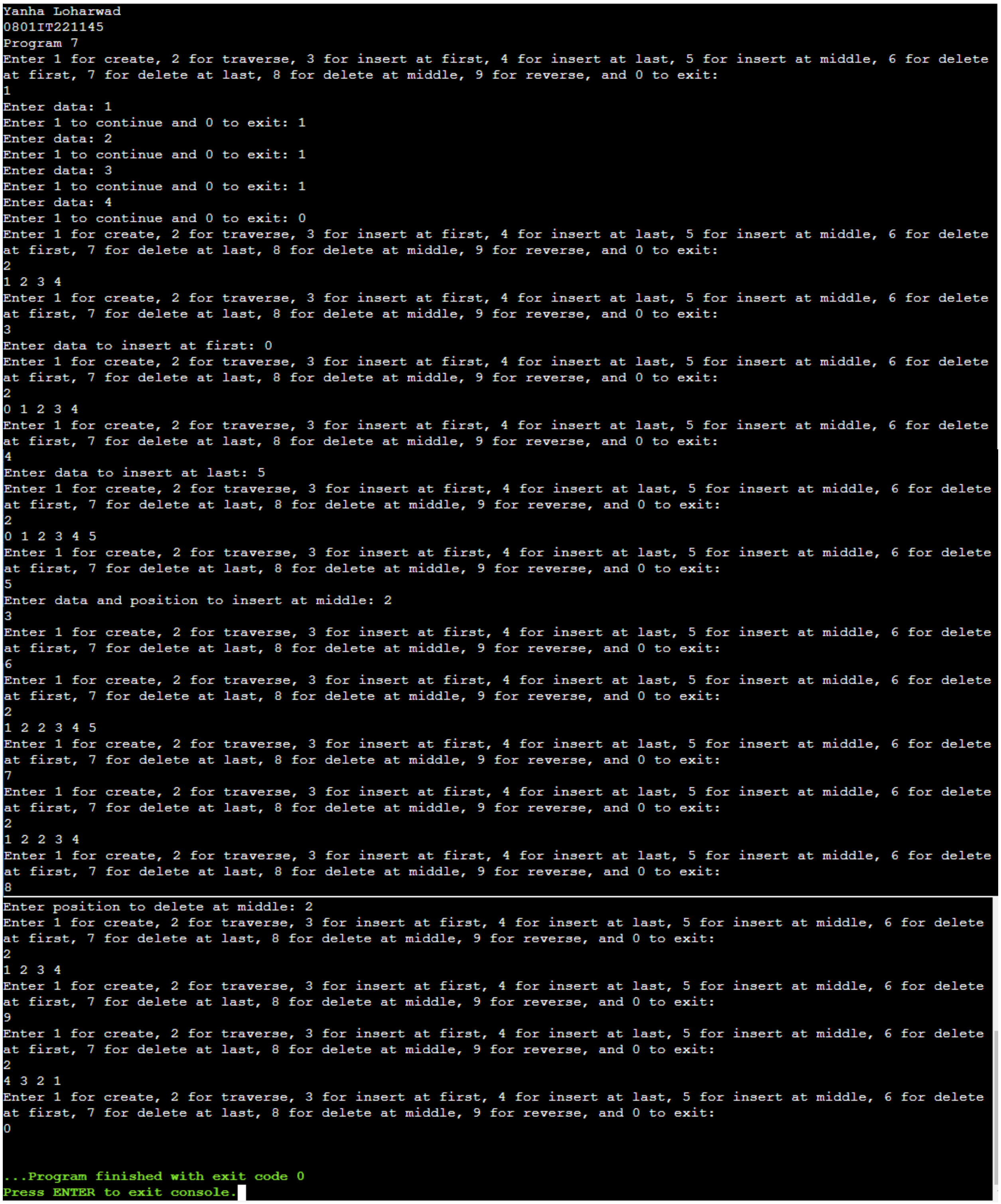
```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
};
struct node *head, *tail, *newnode, *temp;
void create() {
head = NULL;
tail = NULL;
int choice;
do {
newnode = (struct node *)malloc(sizeof(struct node));
printf("Enter data: ");
scanf("%d", &newnode->data);
newnode->next = head;
if (head == NULL) {
head = tail = newnode;
tail->next = head;
} else {
tail->next = newnode;
tail = newnode;
}
printf("Enter 1 to continue and 0 to exit: ");
scanf("%d", &choice);
} while (choice != 0);
}
void traverse() {
if (head == NULL) {
printf("Linked list is empty.\n");
return;
}
temp = head;
do {
printf("%d ", temp->data);
temp = temp->next;
} while (temp != head);
printf("\n");
}
void insertAtFirst(int data) {
newnode = (struct node *)malloc(sizeof(struct node));
newnode->data = data;
newnode->next = head;
tail->next = newnode;
head = newnode;
}
```

```
}
void insertAtLast(int data) {
newnode = (struct node *)malloc(sizeof(struct node));
newnode->data = data;
newnode->next = head;
tail->next = newnode;
tail = newnode;
}
void insertAtMiddle(int data, int position) {
newnode = (struct node *)malloc(sizeof(struct node));
newnode->data = data;
temp = head;
int count = 1;
while (count < position - 1) {
temp = temp->next;
count++;
}
newnode->next = temp->next;
temp->next = newnode;
}
void deleteAtFirst() {
temp = head;
tail->next = temp->next;
head = temp->next;
free(temp);
}
```

```
void deleteAtLast() {
temp = head;
while (temp->next->next != head) {
temp = temp->next;
}
struct node *lastNode = temp->next;
temp->next = head;
free(lastNode);
tail = temp;
}
void deleteAtMiddle(int position) {
temp = head;
int count = 1;
while (count < position - 1) {
temp = temp->next;
count++;
}
struct node *nextnode = temp->next;
temp->next = nextnode->next;
free(nextnode);
}
void reverse() {
struct node *current = head;
struct node *prev = tail;
struct node *next;
```

```
do {
next = current->next;
current->next = prev;
prev = current;
current = next;
} while (current != head);
head = prev;
}
void main() {
printf("Yanha Loharwad\n0801IT221145\nProgram 7\n");
int choice;
head = NULL, tail = NULL;
while (1) {
printf("Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit: \n");
scanf("%d", &choice);
switch (choice) {
case 1: create();  break;
case 2: traverse();break;
case 3: {
int data;
printf("Enter data to insert at first: ");
scanf("%d", &data);
insertAtFirst(data);  break;
}
case 4: {
int data;
printf("Enter data to insert at last: ");
scanf("%d", &data);
insertAtLast(data); break;
}
case 5: {
int data, position;
printf("Enter data and position to insert at middle: ");
scanf("%d %d", &data, &position);
insertAtMiddle(data, position); break;
}
case 6: deleteAtFirst();break;
case 7: deleteAtLast();break;
case 8: {
int position;
printf("Enter position to delete at middle: ");
scanf("%d", &position);
deleteAtMiddle(position); break;
}
case 9:  reverse();break;
case 0: exit(0);
default: printf("Enter a valid choice\n");  }
}
}
```

**OUTPUT: Singly circular linked list successfully created, traversed, node inserted at first, last and any position, node deleted at first, last and at any position and reversed.**



```
Yanha Loharwad
0801IT221145
Program 7
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
1
Enter data: 1
Enter 1 to continue and 0 to exit: 1
Enter data: 2
Enter 1 to continue and 0 to exit: 1
Enter data: 3
Enter 1 to continue and 0 to exit: 1
Enter data: 4
Enter 1 to continue and 0 to exit: 0
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
3
Enter data to insert at first: 0
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
4
Enter data to insert at last: 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
5
Enter data and position to insert at middle: 2
3
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
6
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
7
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
8
Enter position to delete at middle: 2
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
9
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
4 3 2 1
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete
at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
0


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 8

8. **Creation, traversing, insertion at first, insertion at last ,insertion at any position, deletion at first, deletion at last, deletion at any position, reversing of a doubly circular linked list.**

```
#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node* next;

struct node* prev;

};

struct node* head;

struct node* tail;

struct node* newnode;

struct node* temp;

void create() {

head = NULL;

tail = NULL;

int choice;

do {

newnode = (struct node*)malloc(sizeof(struct node));

printf("Enter data: ");

scanf("%d", &newnode->data);

newnode->next = head;

newnode->prev = tail;
if (head == NULL) {

head = tail = newnode;

} else {

tail->next = newnode;

tail = newnode;

}

tail->next = head;

head->prev = tail;

printf("Enter 1 to continue and 0 to exit: ");

scanf("%d", &choice);

} while (choice != 0);

}

void traverse() {

if (head == NULL) {

printf("Doubly linked list is empty.\n");

return;

}

temp = head;

do {

printf("%d ", temp->data);

temp = temp->next;

} while (temp != head);

printf("\n");

}

void insertAtFirst(int data) {
}
```

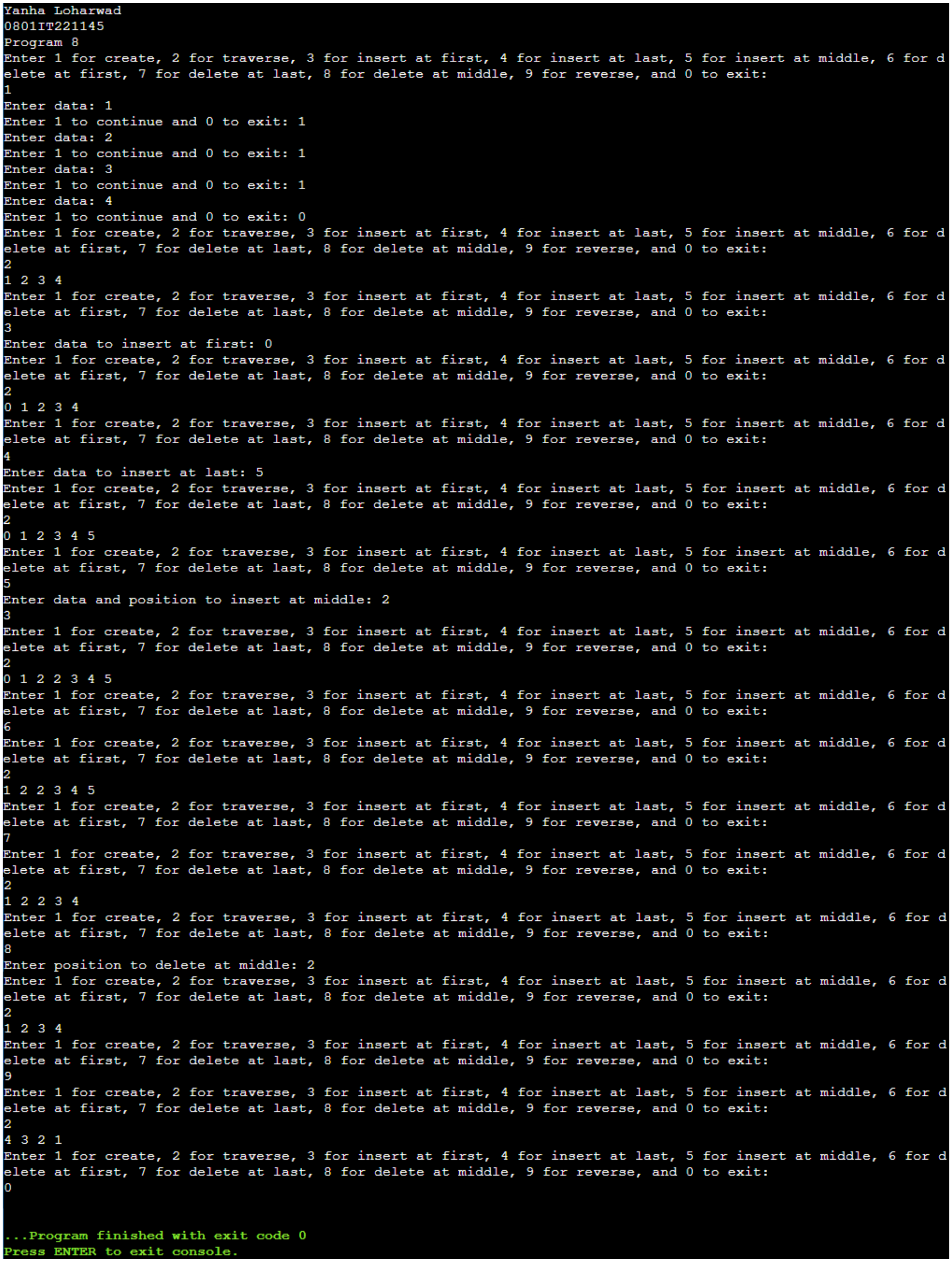
```
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = data;
newnode->next = head;
newnode->prev = tail;
tail->next = newnode;
head->prev = newnode;
head = newnode;
}
void insertAtLast(int data) {
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = data;
newnode->next = head;
newnode->prev = tail;
tail->next = newnode;
tail = newnode;
}
void insertAtMiddle(int data, int position) {
newnode = (struct node*)malloc(sizeof(struct node));
newnode->data = data;
temp = head;
int count = 1;
while (count < position - 1) {
temp = temp->next;
count++;
}
```

```
newnode->next = temp->next;
newnode->prev = temp;
temp->next->prev = newnode;
temp->next = newnode;
}
void deleteAtFirst() {
temp = head;
tail->next = temp->next;
temp->next->prev = tail;
head = temp->next;
free(temp);
}
void deleteAtLast() {
temp = tail;
tail = temp->prev;
tail->next = head;
head->prev = tail;
free(temp);
}
void deleteAtMiddle(int position) {
temp = head;
int count = 1;
while (count < position - 1) {
temp = temp->next;
count++;
```

```
}
struct node* nextnode = temp->next;
temp->next = nextnode->next;
nextnode->next->prev = temp;
free(nextnode);
}
void reverse() {
struct node* current = head;
struct node* tempNode;
do {
tempNode = current->next;
current->next = current->prev;
current->prev = tempNode;
current = tempNode;
} while (current != head);
temp = head;
head = tail;
tail = temp;
}
int main() {
printf("Yanha Loharwad\n0801IT221145\nProgram 8\n");
int choice;
head = NULL;
tail = NULL;
while (1) {
```

```
printf("Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for delete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit: \n");

scanf("%d", &choice);

switch (choice) {

case 1: create(); break;

case 2: traverse(); break;

case 3: {

int data;

printf("Enter data to insert at first: ");

scanf("%d", &data);

insertAtFirst(data);

break;

}

case 4: {

int data;

printf("Enter data to insert at last: ");

scanf("%d", &data);

insertAtLast(data);

break;

}

case 5: {

int data, position;

printf("Enter data and position to insert at middle: ");

scanf("%d %d", &data, &position);

insertAtMiddle(data, position);
break;
}
case 6: deleteAtFirst(); break;
case 7: deleteAtLast(); break;
case 8: {
int position;
printf("Enter position to delete at middle: ");
scanf("%d", &position);
deleteAtMiddle(position);
break;
}
case 9: reverse(); break;
case 0: exit(0);
default: printf("Enter a valid choice\n");
}
}
}
```

**OUTPUT: Doubly circular linked list successfully created, traversed, node inserted at first, last and any position, node deleted at first, last and at any position and reversed.**



```
Yanha Loharwad
0801IT221145
Program 8
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
1
Enter data: 1
Enter 1 to continue and 0 to exit: 1
Enter data: 2
Enter 1 to continue and 0 to exit: 1
Enter data: 3
Enter 1 to continue and 0 to exit: 1
Enter data: 4
Enter 1 to continue and 0 to exit: 0
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
3
Enter data to insert at first: 0
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
4
Enter data to insert at last: 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
5
Enter data and position to insert at middle: 2
3
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
0 1 2 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
6
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 2 3 4 5
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
7
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
8
Enter position to delete at middle: 2
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
1 2 3 4
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
9
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
2
4 3 2 1
Enter 1 for create, 2 for traverse, 3 for insert at first, 4 for insert at last, 5 for insert at middle, 6 for d
elete at first, 7 for delete at last, 8 for delete at middle, 9 for reverse, and 0 to exit:
0


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 9

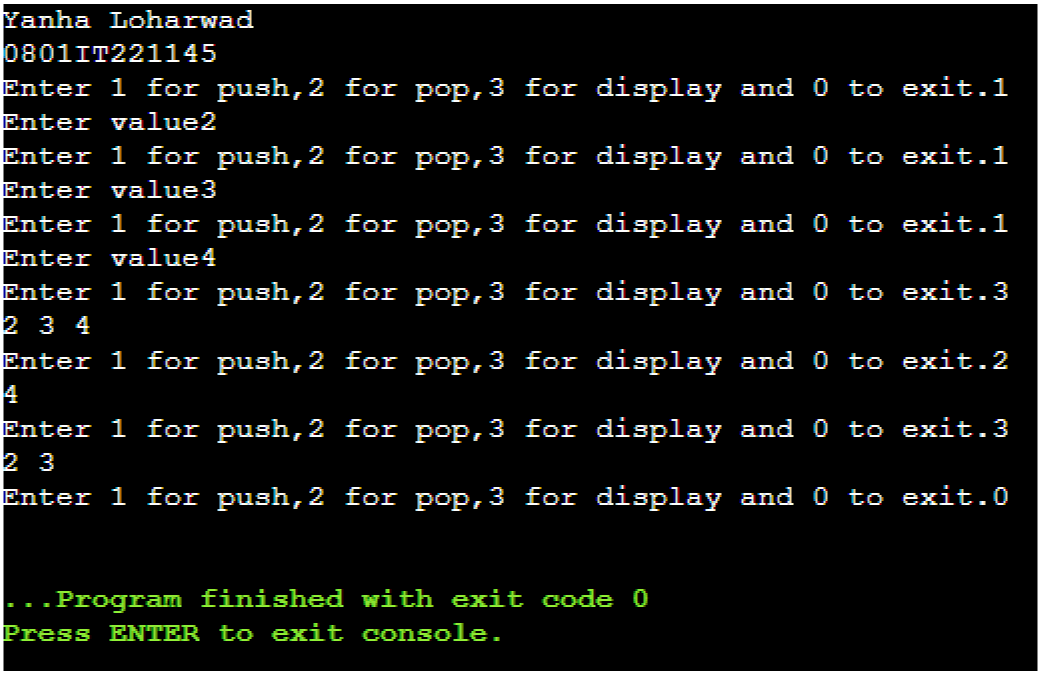
**9. Write a program to implement stack using array.**

```
#include <stdio.h>
#include<stdlib.h>
#define size 5
struct stack{
int A[size];
int top;
}s;
void push(){
if(s.top==size-1)
{
printf("Overflow");
}
else{
int x;
printf("Enter value");
scanf("%d",&x);
s.top++;
s.A[s.top]=x;
}
}
void pop()
{
if(s.top==-1)
```

```
{
printf("Underflow");
}
else{
printf("%d\n",s.A[s.top]);
s.top--;
}
}
void display()
{
if (s.top == -1) {
printf("Underflow\n");
} else {
for (int i = 0; i <= s.top; i++) {
printf("%d ", s.A[i]);
}
printf("\n");
}
}
void main()
{
printf("Yanha Loharwad\n");
printf("0801IT221145\n");
s.top=-1;
int c;
```

```
while(1)

{

printf("Enter 1 for push,2 for pop,3 for display and 0 to exit.");

scanf("%d",&c);

switch(c)

{

case 1:push();  break;

case 2:pop();   break;

case 3:display();break;

case 0:exit(0);

}

}

}
```

**OUTPUT: Stack implemented using array.**



```
Yanha Loharwad
0801IT221145
Enter 1 for push,2 for pop,3 for display and 0 to exit.1
Enter value2
Enter 1 for push,2 for pop,3 for display and 0 to exit.1
Enter value3
Enter 1 for push,2 for pop,3 for display and 0 to exit.1
Enter value4
Enter 1 for push,2 for pop,3 for display and 0 to exit.3
2 3 4
Enter 1 for push,2 for pop,3 for display and 0 to exit.2
4
Enter 1 for push,2 for pop,3 for display and 0 to exit.3
2 3
Enter 1 for push,2 for pop,3 for display and 0 to exit.0


...Program finished with exit code 0
Press ENTER to exit console.
```

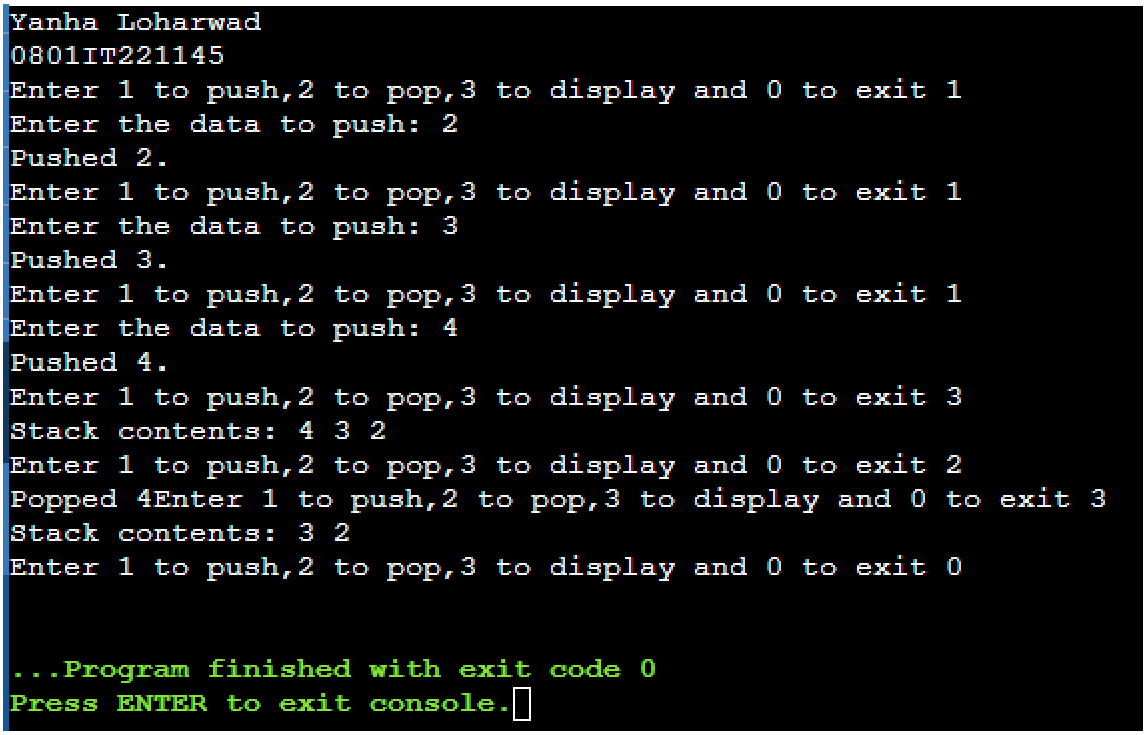
# Program 10

**10. Program to implement stack using linked list.**

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node* next;
};
struct Node* top = NULL;
void push(int value) {
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
if (newNode == NULL) {
printf("Overflow.\n");
return;
}
newNode->data = value;
newNode->next = top;
top = newNode;
printf("Pushed %d.\n", value);
}
void pop() {
if (top == NULL) {
printf("Underflow.\n");
}
```

```
int value = top->data;
struct Node* temp = top;
top = top->next;
free(temp);
printf("Popped %d",value);
}
void display() {
struct Node* current = top;
if (current == NULL) {
printf("Underflow.\n");
} else {
printf("Stack contents: ");
while (current != NULL) {
printf("%d ", current->data);
current = current->next;
}
printf("\n");
}
}
void main() {
printf("Yanha Loharwad\n0801IT221145\n");
int choice, data;
while (1) {
printf("Enter 1 to push,2 to pop,3 to display and 0 to exit ");
scanf("%d", &choice);
switch (choice) {
case 1:  printf("Enter the data to push: ");
scanf("%d", &data);
push(data);
break;
case 2: pop(); break;
case 3: display(); break;
case 0: exit(0);
default: printf("Invalid choice. \n");
}
}
}
```

**OUTPUT: Stack implemented using linked list.**



```
Yanha Loharwad
0801IT221145
Enter 1 to push,2 to pop,3 to display and 0 to exit 1
Enter the data to push: 2
Pushed 2.
Enter 1 to push,2 to pop,3 to display and 0 to exit 1
Enter the data to push: 3
Pushed 3.
Enter 1 to push,2 to pop,3 to display and 0 to exit 1
Enter the data to push: 4
Pushed 4.
Enter 1 to push,2 to pop,3 to display and 0 to exit 3
Stack contents: 4 3 2
Enter 1 to push,2 to pop,3 to display and 0 to exit 2
Popped 4Enter 1 to push,2 to pop,3 to display and 0 to exit 3
Stack contents: 3 2
Enter 1 to push,2 to pop,3 to display and 0 to exit 0


...Program finished with exit code 0
Press ENTER to exit console.
```

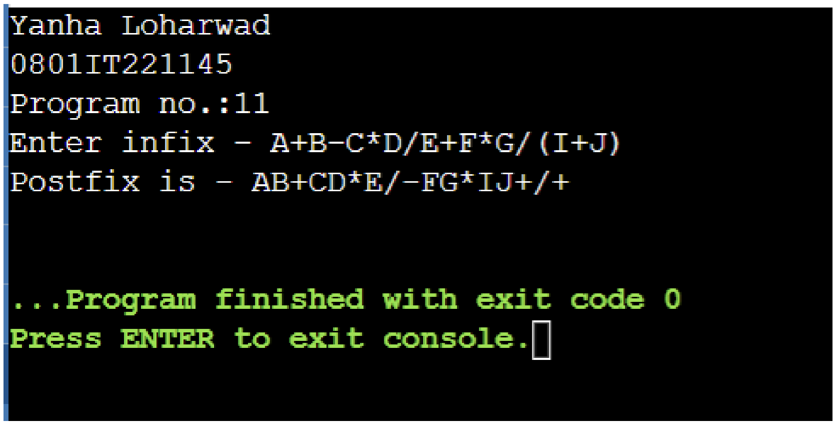
# Program 11

**11.  Infix to postfix conversion using array.**

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 50

struct stack {
int top;
char a[SIZE];
} s;
void push(char elem) {
s.a[++s.top] = elem;
}
char pop() {
return s.a[s.top--];
}

int preci(char ch) {
switch (ch) {
case '#':
return 0;
case '+':
case '-':
return 1;
case '*':
case '/':
return 2;
default:
return 0;
}
}

int main() {
printf("Yanha Loharwad\n0801IT221145\nProgram no.:11\n");
s.top = -1;
push('#');
int i = 0, k = 0;
char infix[SIZE], postfix[SIZE], ch;
```

```
printf("Enter infix - ");
scanf("%s", infix);

while ((ch = infix[i++]) != '\0') {
if (ch == '(') {
push(ch);
} else if (isalnum(ch)) {
postfix[k++] = ch;
} else if (ch == ')') {
while (s.a[s.top] != '(') {
postfix[k++] = pop();
}
char trash = pop();
} else {
while (preci(s.a[s.top]) >= preci(ch)) {
postfix[k++] = pop();
}
push(ch);
}
}
while (s.a[s.top] != '#') {
postfix[k++] = pop();
}
postfix[k] = '\0';
printf("Postfix is - %s\n", postfix);
}
```

**OUTPUT: Infix to postfix converted using array.**



```
Yanha Loharwad
0801IT221145
Program no.:11
Enter infix - A+B-C*D/E+F*G/(I+J)
Postfix is - AB+CD*E/-FG*IJ+/+


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 12

**12. Infix to postfix conversion using linked list.**

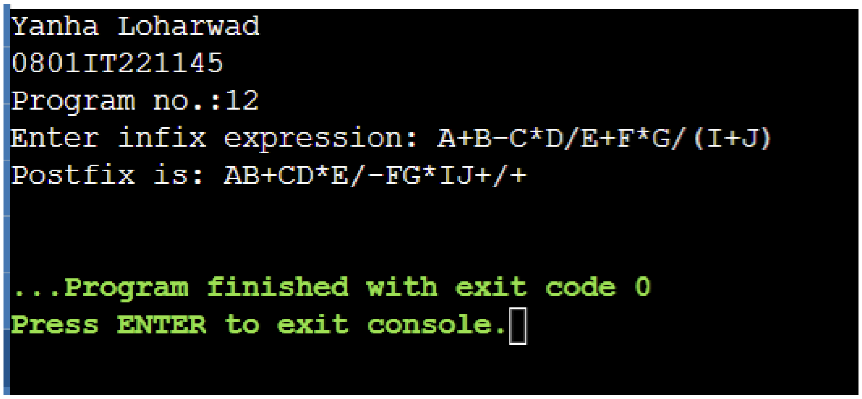
```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

struct Stack {
char* data;
int top;
int size;
};
struct Stack stack;
char* postfix;
int postfixIndex = 0;

void initStack(int stackSize) {
stack.size = stackSize;
stack.data = (char*)malloc(stackSize * sizeof(char));
stack.top = -1;
}
void push(char elem) {
if (stack.top < stack.size - 1) {
stack.data[++stack.top] = elem;
} else {
printf("Error: Stack is full\n");
}
}
char pop() {
if (stack.top >= 0) {
return stack.data[stack.top--];
} else {
printf("Error: Stack is empty\n");
return '\0';
}
}
int precedence(char op) {
switch (op) {
case '+':
case '-':
return 1;
case '*':
case '/':
return 2;
default:
return 0;
}
}
void infixToPostfix(const char* infix) {
```

}

```
initStack(strlen(infix)); // Initialize the stack with a dynamic size
postfix = (char*)malloc(strlen(infix) * sizeof(char));
postfixIndex = 0;
int i = 0;
while (infix[i] != '\0') {
char ch = infix[i];
if (isalnum(ch)) {
postfix[postfixIndex++] = ch;
} else if (ch == '(') {
push(ch);
} else if (ch == ')') {
while (stack.top >= 0 && stack.data[stack.top] != '(') {
postfix[postfixIndex++] = pop();
}
if (stack.top >= 0 && stack.data[stack.top] == '(') {
pop();
}
} else {
while (stack.top >= 0 && precedence(stack.data[stack.top]) >= precedence(ch)) {
postfix[postfixIndex++] = pop();
}push(ch);
}
i++;
}
while (stack.top >= 0) {
```

```
postfix[postfixIndex++] = pop();

}

postfix[postfixIndex] = '\0';

printf("Postfix is: %s\n", postfix);

free(stack.data);

free(postfix);

}


void main() {

printf("Yanha Loharwad\n0801IT221145\nProgram no.:12\n");

char infix[50];

printf("Enter infix expression: ");

scanf("%s", infix);

infixToPostfix(infix);

}
```

**OUTPUT: Infix to postfix converted using linked list.**



```
Yanha Loharwad
0801IT221145
Program no.:12
Enter infix expression: A+B-C*D/E+F*G/(I+J)
Postfix is: AB+CD*E/-FG*IJ+/+


...Program finished with exit code 0
Press ENTER to exit console.
```

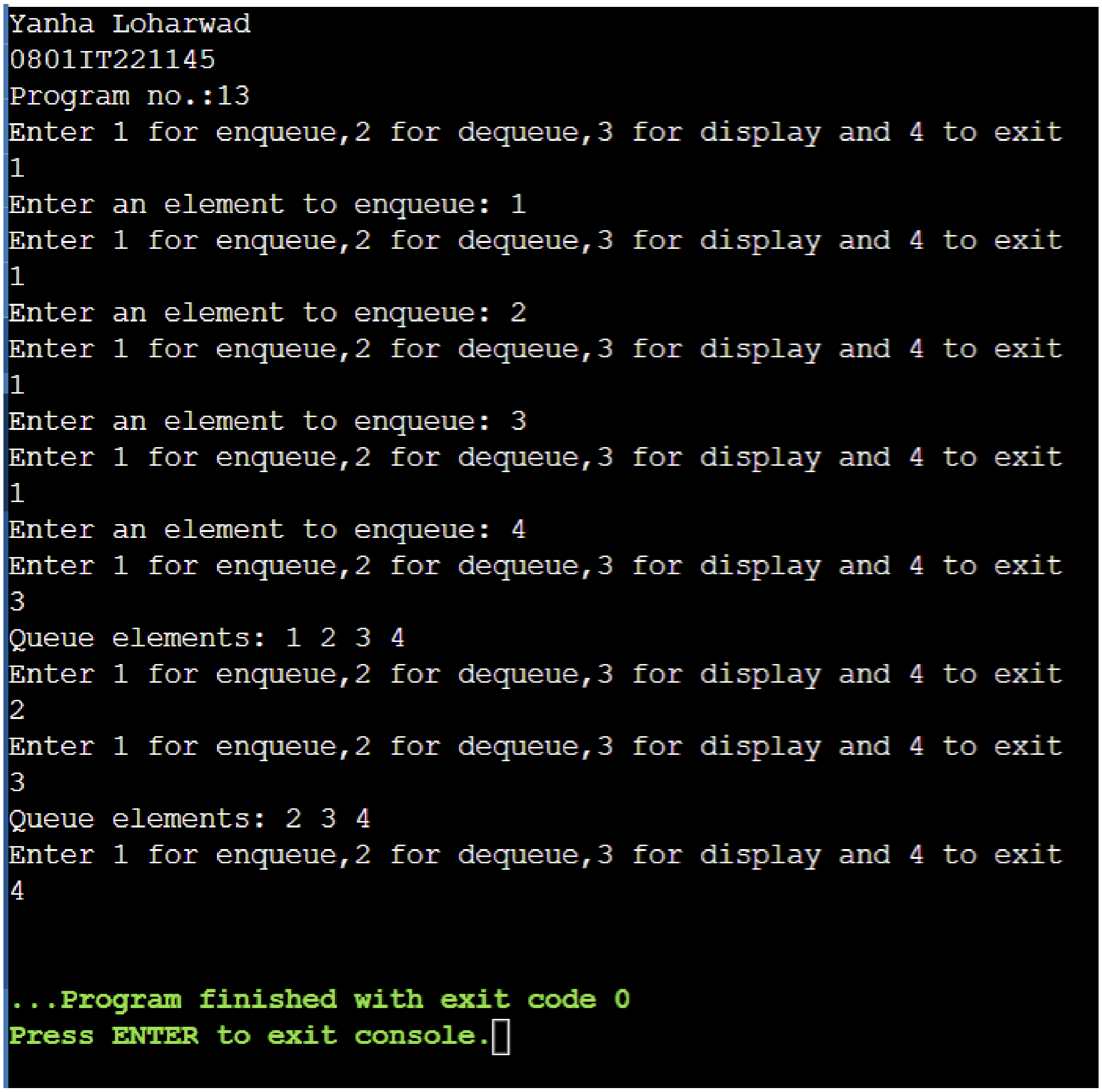
# Program 13

**13. Implementation of queue using array.**

```
#include <stdio.h>
#include <stdlib.h>
#define size 100

struct Queue {
int array[size];
int front;
int rear;
} q;

void enqueue(int data) {
if (q.rear == size - 1) {
printf("Overflow.\n");
} else {
if (q.front == -1) {
q.front = 0;
}
q.rear++;
q.array[q.rear] = data;
}
}

void dequeue() {
if (q.front == -1) {
printf("Underflow.\n");
} else {
int dequeuedItem = q.array[q.front];
if (q.front == q.rear) {
q.front = q.rear = -1;
} else {
q.front++;
}
}
}

void display() {
```

```
if (q.front == -1) {
printf("Underflow.\n");
return;
}

printf("Queue elements: ");
for (int i = q.front; i <= q.rear; i++) {
printf("%d ", q.array[i]);
}
printf("\n");
}

int main() {
q.front = -1;
q.rear = -1;
int choice;
int data;
printf("Yanha Loharwad\n0801IT221145\nProgram no.:13\n");
while (1) {
printf("Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit\n");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter an element to enqueue: ");
scanf("%d", &data);
enqueue(data);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 4:
exit(0);
default:
printf("Invalid choice.\n");
}
}
}
```

**OUTPUT: Queue implemented successfully using array.**



```
Yanha Loharwad
0801IT221145
Program no.:13
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
1
Enter an element to enqueue: 1
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
1
Enter an element to enqueue: 2
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
1
Enter an element to enqueue: 3
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
1
Enter an element to enqueue: 4
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
3
Queue elements: 1 2 3 4
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
2
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
3
Queue elements: 2 3 4
Enter 1 for enqueue,2 for dequeue,3 for display and 4 to exit
4


...Program finished with exit code 0
Press ENTER to exit console.
```

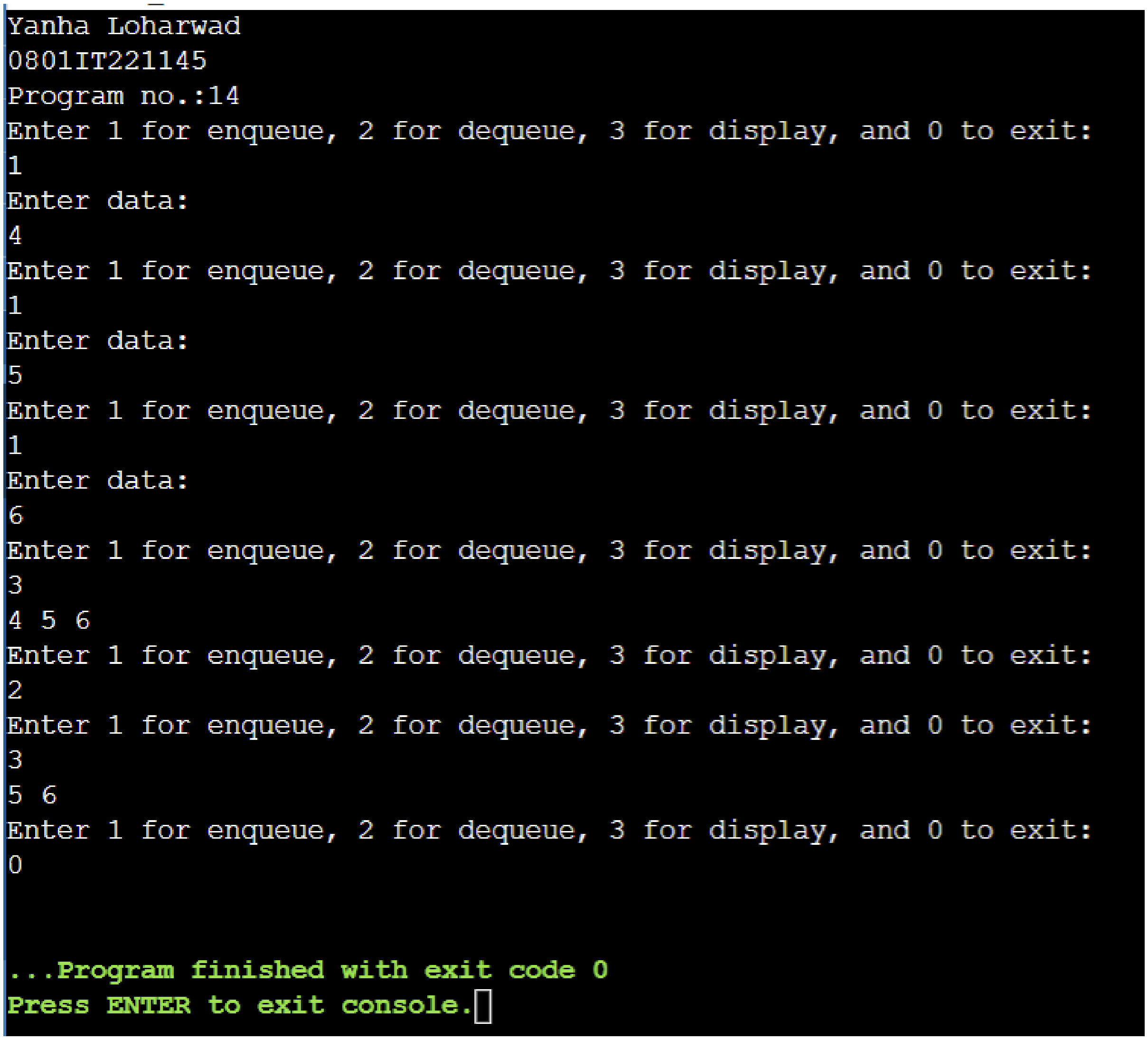
# Program 14

**14. Implementation of queue using linked list.**

```
#include <stdio.h>
#include <stdlib.h>

struct q {
int data;
struct q *next;
};

struct q *rear, *newnode,*temp;

void enqueue(int data) {
newnode = (struct q*)malloc(sizeof(struct q));
newnode->data = data;
newnode->next = NULL;
if (rear == NULL) {
rear = newnode;
} else {
temp = rear;
while (temp->next != NULL) {
temp = temp->next;
}
temp->next = newnode;
}
}

void dequeue() {
if (rear == NULL) {
printf("Underflow.\n");
return;
}
temp = rear;
rear = rear->next;
free(temp);
}

void display() {
```

```
temp = rear;
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}

int main() {
printf("Yanha Loharwad\n0801IT221145\nProgram no.:14\n");
int choice;
rear = NULL;
while (1) {
printf("Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: \n");
scanf("%d", &choice);
switch (choice) {
case 1:
int data;
printf("Enter data:\n");
scanf("%d", &data);
enqueue(data);
break;
case 2:
dequeue();
break;
case 3:
if (rear == NULL) {
printf("Linked list is empty.\n");
} else {
display();
}
break;
case 0:
exit(0);
default:

printf("Enter a valid choice\n");
}
}
}
```

**OUTPUT: Queue implemented successfully using linked list.**



```
Yanha Loharwad
0801IT221145
Program no.:14
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
1
Enter data:
4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
1
Enter data:
5
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
1
Enter data:
6
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
3
4 5 6
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
2
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
3
5 6
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
0


...Program finished with exit code 0
Press ENTER to exit console.
```

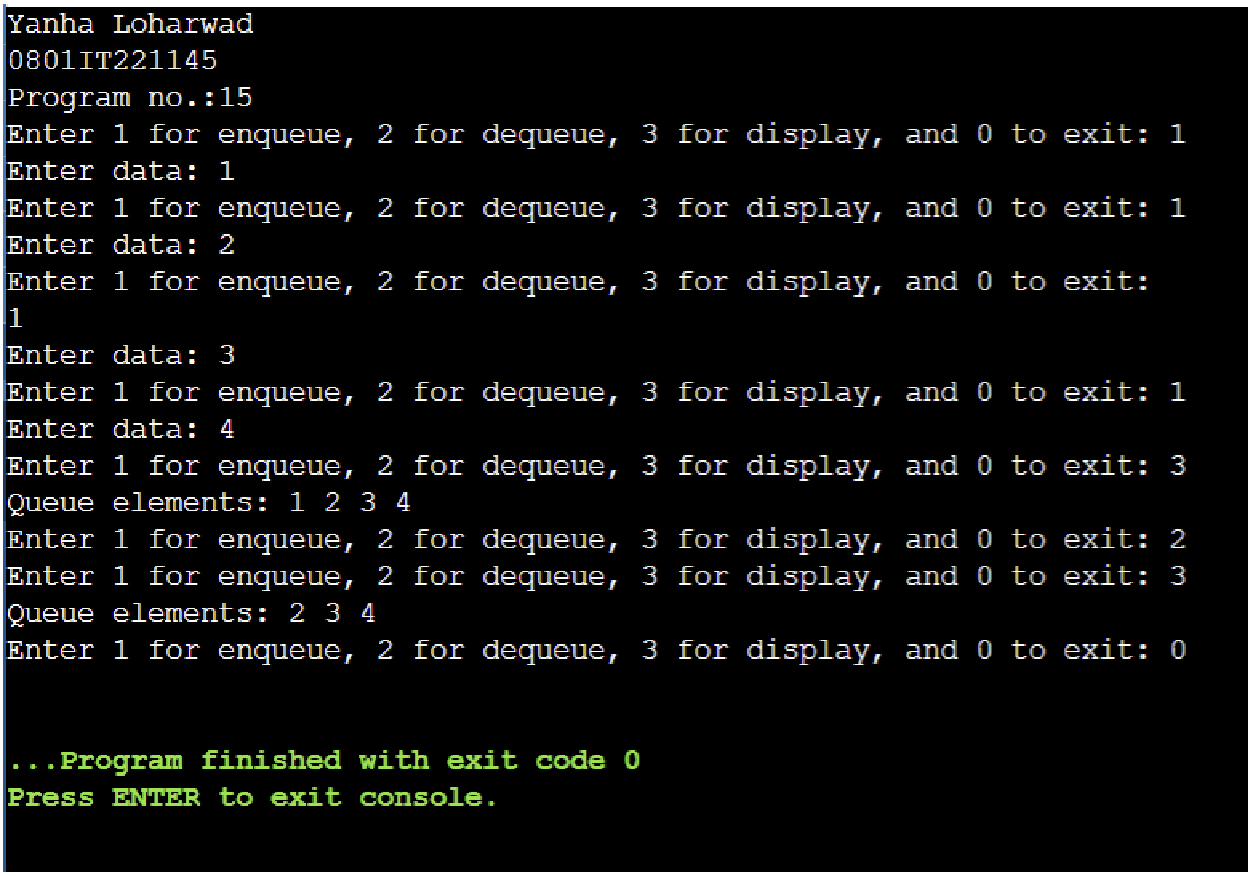
# Program 15

**15. Implementation of circular queue using array.**

```
#include <stdio.h>
#include <stdlib.h>
#define size 100
int queue[size];
int front = -1;
int rear = -1;

void enqueue(int data) {
if ((front == 0 && rear == size - 1) || (rear == front - 1)) {
printf("Queue is full. Cannot enqueue.\n");
} else {
if (front == -1) {
front = 0;
}
rear = (rear + 1) % size;
queue[rear] = data;
}
}

void dequeue() {
if (front == -1) {
printf("Queue is empty. Cannot dequeue.\n");
} else {
if (front == rear) {
front = rear = -1;
} else {
front = (front + 1) % size;
}
}
}

void display() {
int i;
if (front == -1) {
printf("Queue is empty.\n");
} else {
printf("Queue elements: ");
for (i = front; i != rear; i = (i + 1) % size) {
printf("%d ", queue[i]);
}
printf("%d\n", queue[i]);
}
}

void main() {
int choice, data;
printf("Yanha Loharwad\n0801IT221145\nProgram no.:15\n");
```

```
while (1) {
printf("Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter data: ");
scanf("%d", &data);
enqueue(data);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
case 0:
exit(0);
default:
printf("Invalid choice.\n");
}
}
}
```

**OUTPUT: Circular queue implemented successfully using array.**



```
Yanha Loharwad
0801IT221145
Program no.:15
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 1
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 2
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit:
1
Enter data: 3
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 3
Queue elements: 1 2 3 4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 2
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 3
Queue elements: 2 3 4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 0


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 16

**16. Implementation of circular queue using linked list.**

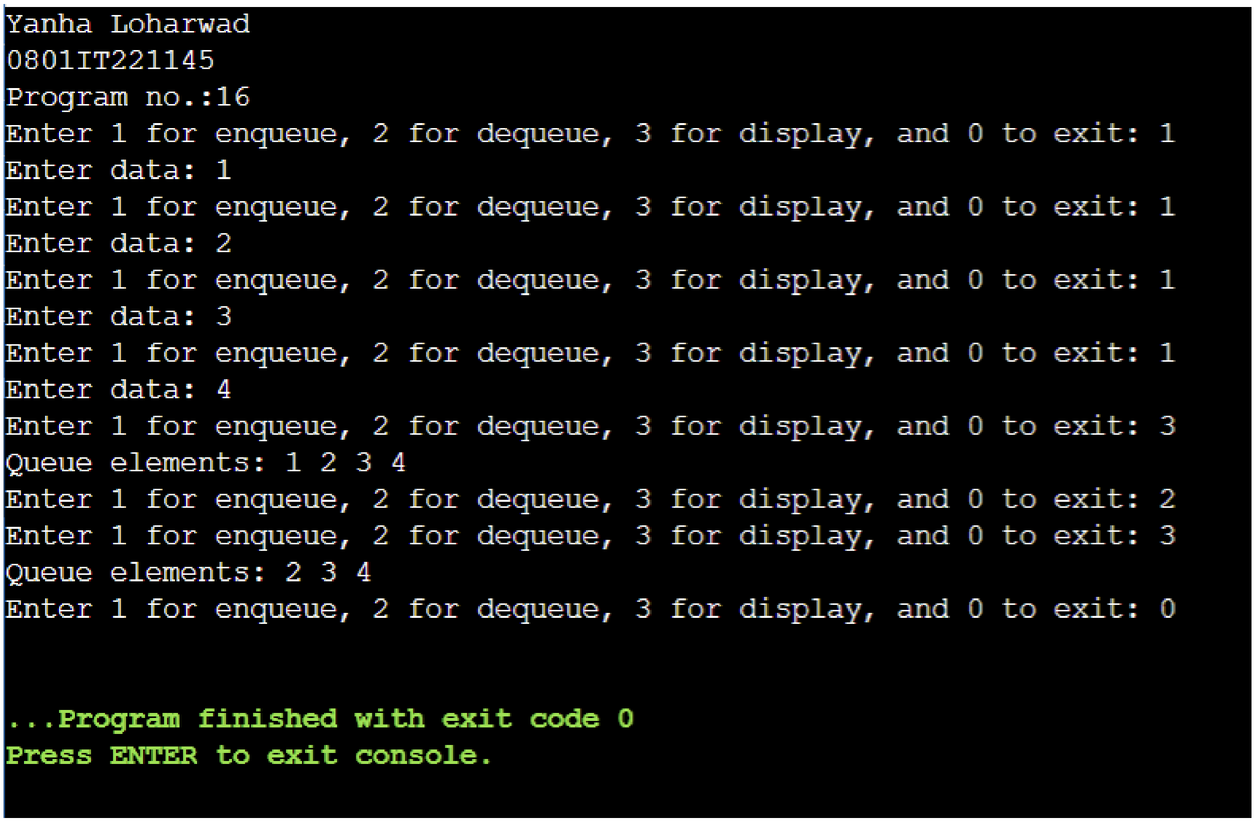
```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node* next;
};
struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int data) {
struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
newnode->data = data;
newnode->next = NULL;

if (front == NULL) {
front = newnode;
} else {
rear->next = newnode;
}
rear = newnode;
rear->next = front; // Make it circular
}
```

```
void dequeue() {
if (front == NULL) {
printf("Queue is empty. Cannot dequeue.\n");
return;
}

if (front == rear) {
free(front);
front = rear = NULL;
} else {
struct Node* temp = front;
front = front->next;
rear->next = front; // Make it circular
free(temp);
}
}

void display() {
if (front == NULL) {
printf("Queue is empty.\n");
} else {
struct Node* current = front;
printf("Queue elements: ");
do {
```

```
printf("%d ", current->data);
current = current->next;
} while (current != front);
printf("\n");
}
}

int main() {
int choice, data;
printf("Yanha Loharwad\n0801IT221145\nProgram no.:16\n");
while (1) {
printf("Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter data: ");
scanf("%d", &data);
enqueue(data);
break;
case 2:
dequeue();
break;
case 3:
display();
break;
```

```
case 0:

exit(0);

default:

printf("Invalid choice.\n");

}

}

}
```

**OUTPUT: Circular queue implemented successfully using linked list.**



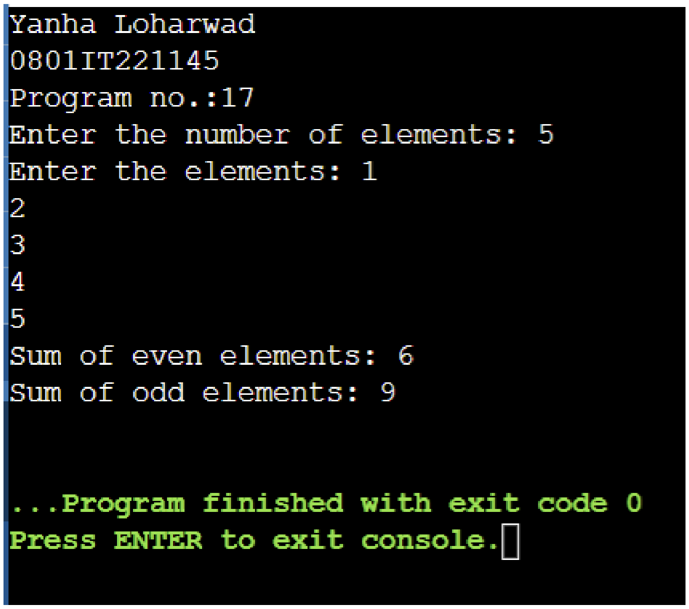
```
Yanha Loharwad
0801IT221145
Program no.:16
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 1
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 2
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 3
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 1
Enter data: 4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 3
Queue elements: 1 2 3 4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 2
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 3
Queue elements: 2 3 4
Enter 1 for enqueue, 2 for dequeue, 3 for display, and 0 to exit: 0


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 17

**17. Write a program to print sum of even and odd elements using malloc() and free().**

```
#include <stdio.h>
#include <stdlib.h>

int main() {
int n;
printf("Yanha Loharwad\n0801IT221145\nProgram no.:17\nEnter the number of elements: ");
scanf("%d", &n);
int *arr = (int *)malloc(n * sizeof(int));
printf("Enter the elements: ");
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
int evenSum = 0, oddSum = 0;
for (int i = 0; i < n; i++) {
if (arr[i] % 2 == 0) {
evenSum += arr[i];
} else {
oddSum += arr[i];
}
}
printf("Sum of even elements: %d\n", evenSum);
printf("Sum of odd elements: %d\n", oddSum);
free(arr);

return 0;

}
```

**OUTPUT:Even and odd numbers sum is calculated.**



```
Yanha Loharwad
0801IT221145
Program no.:17
Enter the number of elements: 5
Enter the elements: 1
2
3
4
5
Sum of even elements: 6
Sum of odd elements: 9


...Program finished with exit code 0
Press ENTER to exit console.
```

# Program 18

**18. Write a program to merge linked list.**

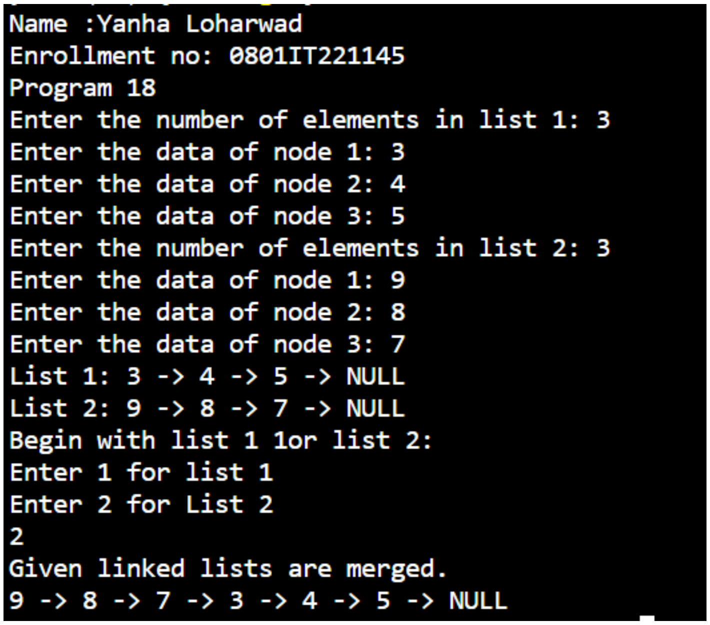
```
#include <stdio.h>
#include <stdlib.h>

struct node {
int data;
struct node *next;
};
void createList(struct node **head, int n) {
int data;
struct node *newnode = NULL, *temp = NULL;
*head = (struct node *)malloc(sizeof(struct node));
if (*head == NULL) {
printf("Memory not allocated.\n");
exit(1);
}
printf("Enter the data of node 1: ");
scanf("%d", &data);
(*head)->data = data;
(*head)->next = NULL;
temp = *head;
for (int i = 1; i < n; i++) {
newnode = (struct node *)malloc(sizeof(struct node));
```

```
if (newnode == NULL) {
printf("Memory not allocated.\n");
exit(1);
}
printf("Enter the data of node %d: ", i + 1);
scanf("%d", &data);
newnode->data = data;
newnode->next = NULL;
temp->next = newnode;
temp = temp->next;
}
}
void displayList(struct node *head) {
struct node *current = head;
while (current != NULL) {
printf("%d -> ", current->data);
current = current->next;
}
printf("NULL\n");
}
void mergeLists(struct node **head1 , struct node **head2){
struct node * temp;
temp =(*head1);
while(temp->next!=NULL){
temp=temp->next;
```

```
}
temp->next=(*head2);
printf("Given linked lists are merged.\n");
displayList((*head1));
}

int main() {
printf("Name :Yanha Loharwad\nEnrollment no: 0801IT221145\nProgram 18\n");
int n, m,c;
struct node *head1 = NULL, *head2 = NULL;
printf("Enter the number of elements in list 1: ");
scanf("%d", &n);
createList(&head1, n);
printf("Enter the number of elements in list 2: ");
scanf("%d", &m);
createList(&head2, m);
printf("List 1: ");
displayList(head1);
printf("List 2: ");
displayList(head2);
printf("Begin with list 1 1or list 2:\nEnter 1 for list 1\nEnter 2 for List 2\n");
scanf("%d",&c);
switch(c){
case 1 : mergeLists(&head1,&head2);
break;
```

```
case 2 : mergeLists(&head2,&head1);

break;

default :

printf("Enter valid option.\n");

break;

}

return 0;

}
```

**OUTPUT: Linked lists are successfully merged.**



```
Name :Yanha Loharwad
Enrollment no: 0801IT221145
Program 18
Enter the number of elements in list 1: 3
Enter the data of node 1: 3
Enter the data of node 2: 4
Enter the data of node 3: 5
Enter the number of elements in list 2: 3
Enter the data of node 1: 9
Enter the data of node 2: 8
Enter the data of node 3: 7
List 1: 3 -> 4 -> 5 -> NULL
List 2: 9 -> 8 -> 7 -> NULL
Begin with list 1 1or list 2:
Enter 1 for list 1
Enter 2 for List 2
2
Given linked lists are merged.
9 -> 8 -> 7 -> 3 -> 4 -> 5 -> NULL
```

# Program 19

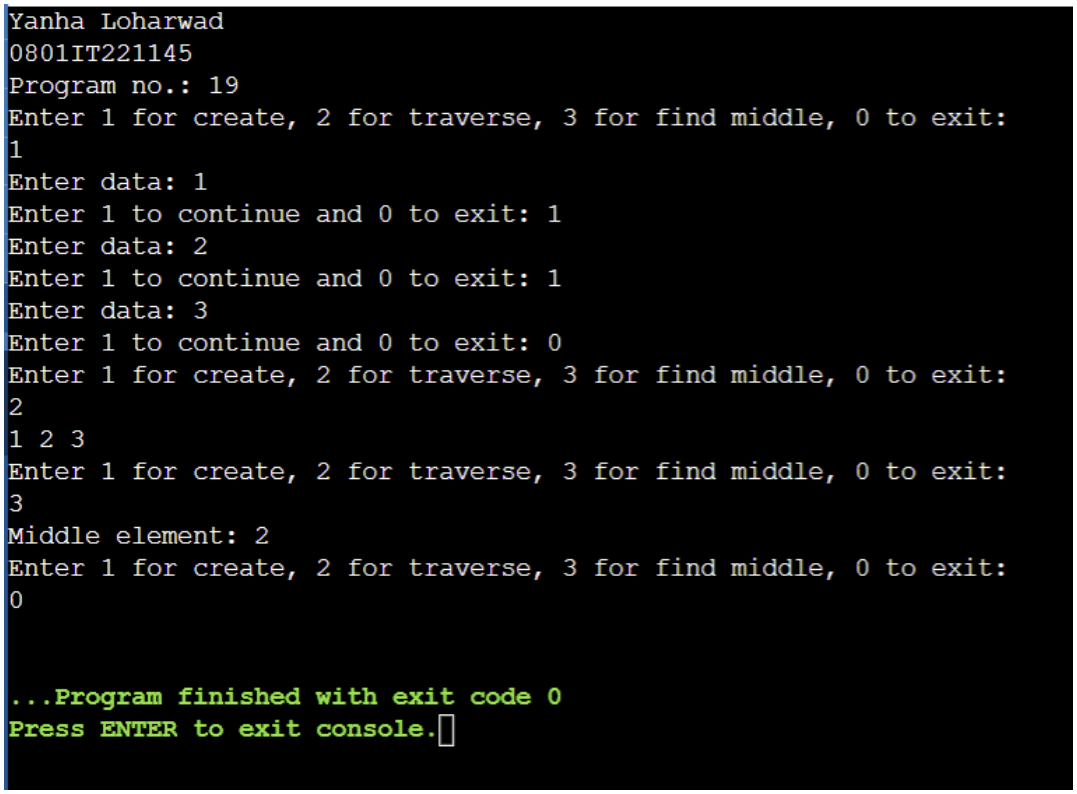
**19. Write a program to merge linked list.**

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *head, *newnode, *temp;
void create()
{
head = NULL;
int choice;
do
{
newnode = (struct node *)malloc(sizeof(struct node));
printf("Enter data: ");
scanf("%d", &newnode->data);
newnode->next = NULL;
if (head == NULL)
{
head = temp = newnode;
}
else

{

temp->next = newnode;

temp = newnode;

}

printf("Enter 1 to continue and 0 to exit: ");

scanf("%d", &choice);

} while (choice != 0);

}

void traverse()

{

temp = head;

while (temp != NULL)

{

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

struct node* findMiddle(struct node* head)

{

if (head == NULL)

{

return NULL;

}
```

```
struct node* current = head;

struct node* middle = head;

int count = 0;


while (current != NULL)

{

if (count % 2 == 1)

{

middle = middle->next;

}

current = current->next;

count++;

}

return middle;

}


int main()

{

printf("Yanha Loharwad\n0801IT221145\nProgram no.: 19\n");

int choice;

head = NULL;

while (1)

{

printf("Enter 1 for create, 2 for traverse, 3 for find middle, 0 to exit: \n");

scanf("%d", &choice);
```

```
switch (choice)
{
case 1:
create();
break;
case 2:
if (head == NULL)
{
printf("Linked list is empty. \n");
}
else
{
traverse();
}
break;
case 3:
if (head == NULL)
{
printf("Linked list is empty. \n");
}
else
{
struct node* middle = findMiddle(head);
printf("Middle element: %d\n", middle->data);
}
break;

case 0:

exit(0);

default:

printf("Enter a valid choice\n");

}

}

}
```

**OUTPUT:Middle element of the array is successfully found.**



```
Yanha Loharwad
0801IT221145
Program no.: 19
Enter 1 for create, 2 for traverse, 3 for find middle, 0 to exit:
1
Enter data: 1
Enter 1 to continue and 0 to exit: 1
Enter data: 2
Enter 1 to continue and 0 to exit: 1
Enter data: 3
Enter 1 to continue and 0 to exit: 0
Enter 1 for create, 2 for traverse, 3 for find middle, 0 to exit:
2
1 2 3
Enter 1 for create, 2 for traverse, 3 for find middle, 0 to exit:
3
Middle element: 2
Enter 1 for create, 2 for traverse, 3 for find middle, 0 to exit:
0


...Program finished with exit code 0
Press ENTER to exit console.
```

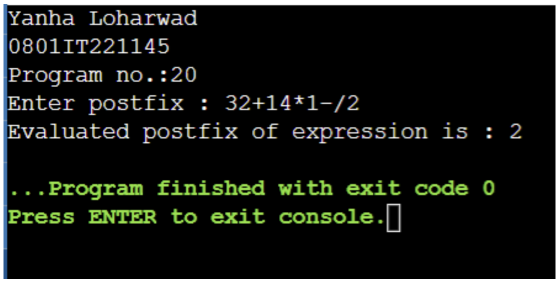
# Program 20

**20. Write a program to evaluate postfix expression.**

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 10
struct stack
{
int top;
int a[SIZE];
} s;
void push(int elem)
{
s.a[++s.top] = elem;
}
int pop()
{
return s.a[s.top--];
}
int isOperand(char x)
{
if (x == '+' || x == '-' || x == '*' || x == '/')
{
return 0;
}
else
return 1;
}
int eval(char *postfix)
{
int i = 0;
int x1, x2, r = 0;
for (i = 0; postfix[i] != '\0'; i++)
{
if (isOperand(postfix[i]))
{
push(postfix[i] - '0');
}
else
{
```

```
x2 = pop();
x1 = pop();
switch (postfix[i])
{
case '+':
r = x1 + x2;
break;
case '-':
r = x1 - x2;
break;
case '*':
r = x1 * x2;
break;
case '/':
r = x1 / x2;
break;
}
push(r);
}
}
return s.a[s.top];
}
int main()
{
printf("Yanha Loharwad\n0801IT221145\nProgram no.:20\n");
s.top = -1;
char postfix[15];
printf("Enter postfix : ");
scanf("%s", postfix);
eval(postfix);
printf("Evaluated postfix of expression is : %d",s.a[s.top]);
}
```

**OUTPUT: Postfix expression evaluated.**



```
Yanha Loharwad
0801IT221145
Program no.:20
Enter postfix : 32+14*1-/2
Evaluated postfix of expression is : 2

...Program finished with exit code 0
Press ENTER to exit console.
```