**LAB 7**

**1/5/23**

**Q1) Implement doubly linked list.**

**PROGRAM CODE:**

//circular doubly linked list//

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct node

{

int data;

struct node\* next;

struct node\* prev;

};

struct node\* head;

int insertbeg(int data)

{

struct node\* newnode;

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

newnode->next = newnode->prev = newnode;

newnode->data = data;

if(head == NULL)

{

head = newnode;

head->next = head->prev = head;

}

else

{

newnode->next = head;

newnode->prev = head->prev;

head->prev->next = newnode;

head->prev = newnode;

head = newnode;

}

return 0;

}

int insertend(int data)

{

struct node\* newnode;

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

newnode->next = newnode->prev = newnode;

newnode->data = data;

struct node\* temp = head;

if(head == NULL)

{

head = newnode;

head->next = head->prev = head;

}

else

{

while(temp->next != head)

{

temp = temp->next;

}

temp->next = newnode;

newnode->prev = temp;

newnode->next = head;

head->prev = newnode;

}

return 0;

}

int insertloc(int loc, int data)

{

struct node\* newnode;

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

newnode->next = newnode->prev = newnode;

newnode->data = data;

struct node\* temp = head;

for(int i=1; i<loc-1; i++)

{

temp = temp->next;

}

newnode->next = temp->next;

temp->next->prev = newnode;

newnode->prev = temp;

temp->next = newnode;

return 0;

}

int deletebeg()

{

if(head == NULL)

{

printf("NOTHING PRESENT\n");

}

else

{

struct node\* temp = head;

head->prev->next = head->next;

head->next->prev = head->prev;

head = head->next;

}

return 0;

}

int deleteend()

{

if(head == NULL)

{

printf("NOTHING PRESENT\n");

}

else

{

struct node\* temp = head;

while(temp->next != head)

{

temp = temp->next;

}

temp->prev->next = head;

head->prev = temp->prev;

free(temp);

}

return 0;

}

int deleteloc(int loc)

{

struct node\* temp= head;

struct node\* ptr;

for(int i=1; i<loc; i++)

{

ptr = temp;

temp = temp->next;

}

ptr->next = temp->next;

temp->next->prev = ptr;

free(temp);

return 0;

}

void traverse()

{

struct node\* temp = head;

while(temp->next!=head)

{

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

temp = temp->next;

}

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

}

void main()

{

int ch=1;

while(ch==1)

{

int choice,d;

printf("Enter 1 for Insertbeg, 2 for Insertend, 3 for Insertloc\n");

printf("Enter 4 for deletebeg, 5 for Deleteend, 6 for Deleteloc\n");

printf("Enter 7 for traversal\n");

scanf("%d",&choice);

switch (choice)

{

case 1:

{

printf("Input Data\n");

scanf("%d",&d);

insertbeg(d);

break;

}

case 2:

{

printf("Input Data\n");

scanf("%d",&d);

insertend(d);

break;

}

case 3:

{

int l;

printf("Enter The location\n");

scanf("%d",&l);

printf("Input Data\n");

scanf("%d",&d);

insertloc(l,d);

break;

}

case 4:

{

deletebeg();

break;

}

case 5:

{

deleteend();

break;

}

case 6:

{

int loc;

printf("Enter The location\n");

scanf("%d",&loc);

deleteloc(loc);

break;

}

case 7:

{

traverse();

}

}

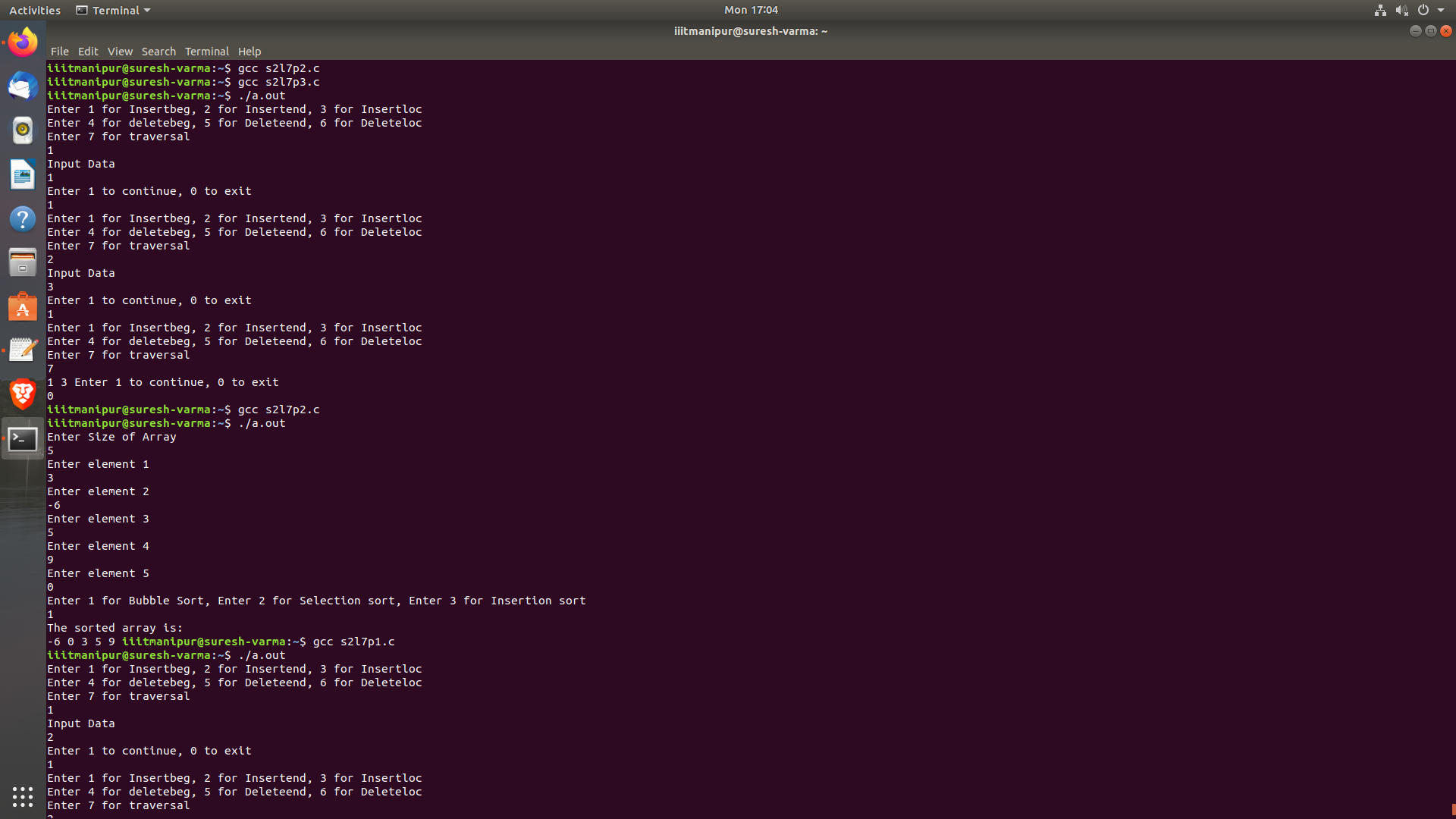
printf("Enter 1 to continue, 0 to exit\n");

scanf("%d",&ch);

}

}

**OUTPUT:**



**Q2) Implement Bubble Sorting Algorithm.**

**PROGRAM CODE:**

#include<stdio.h>

void main()

{

int n;

int array[10];

printf("Enter Size of Array\n");

scanf("%d",&n);

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

{

printf("Enter element %d\n",i+1);

scanf("%d",&array[i]);

}

int ch;

printf("Enter 1 for Bubble Sort, Enter 2 for Selection sort, Enter 3 for Insertion sort\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

{

//Bubble Sort//

int temp;

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

{

for(int j=0; j<n; j++)

{

if(array[j]>array[j+1])

{

temp = array[j];

array[j] = array[j+1];

array[j+1] = temp;

}

}

}

break;

}

case 2:

{

//Selection sort//

int temp;

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

{

for(int j=i; j<=n; j++)

{

if(array[j]<array[i])

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

break;

}

case 3:

{

//Insertion Sort//

int j;

int temp;

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

{

temp = array[i];

j = i-1;

while(j>=0 && array[j] > temp)

{

array[j+1] = array[j];

j--;

}

array[j+1] = temp;

}

break;

}

}

printf("The sorted array is:\n");

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

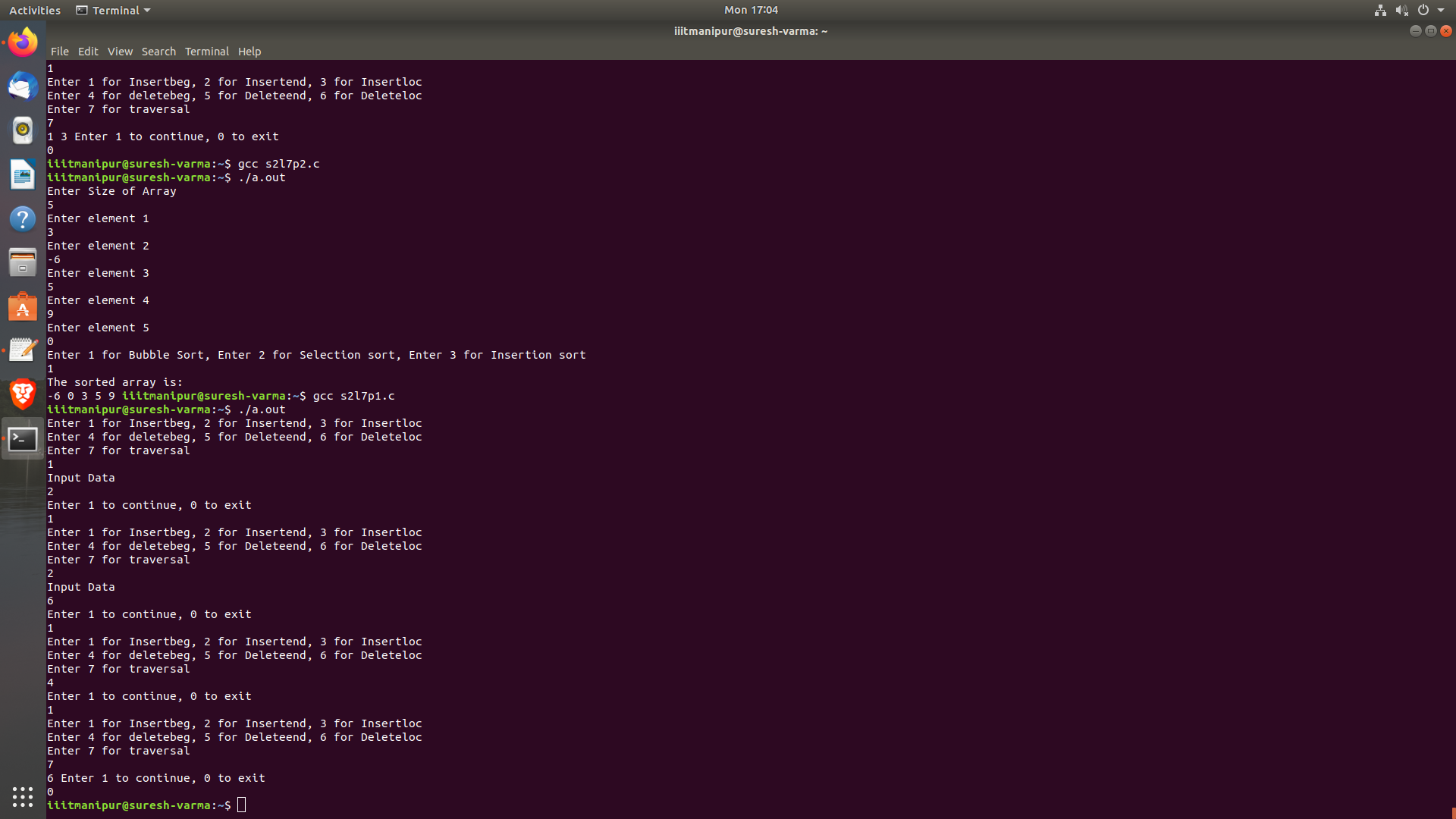
{

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

}

}

**OUTPUT:**



**Q3) Implement a circular doubly linked list.**

**PROGRAM CODE:**

//doubly linked list//

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct node

{

int data;

struct node\* next;

struct node\* prev;

};

struct node\* head;

int insertbeg(int data)

{

struct node\* newnode;

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

newnode->next = newnode->prev = NULL;

newnode->data = data;

if(head == NULL)

{

head = newnode;

head->next = head->prev = NULL;

}

else

{

newnode->next = head;

head->prev=newnode;

head = newnode;

}

return 0;

}

int insertend(int data)

{

struct node\* newnode;

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

newnode->next = newnode->prev = newnode;

newnode->data = data;

struct node\* temp = head;

if(head == NULL)

{

head = newnode;

head->next = head->prev = NULL;

}

else

{

while(temp->next != NULL)

{

temp = temp->next;

}

temp->next = newnode;

newnode->prev = temp;

newnode->next = NULL;

}

return 0;

}

int insertloc(int loc, int data)

{

struct node\* newnode;

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

newnode->next = newnode->prev = newnode;

newnode->data = data;

struct node\* temp = head;

for(int i=1; i<loc-1; i++)

{

temp = temp->next;

}

newnode->next = temp->next;

temp->next->prev = newnode;

newnode->prev = temp;

temp->next = newnode;

return 0;

}

int deletebeg()

{

if(head == NULL)

{

printf("NOTHING PRESENT\n");

}

else

{

struct node\* temp = head;

head = head->next;

free(temp);

head->prev = NULL;

}

return 0;

}

int deleteend()

{

if(head == NULL)

{

printf("NOTHING PRESENT\n");

}

else

{

struct node\* temp = head;

struct node\* ptr;

while(temp->next != NULL)

{

ptr = temp;

temp = temp->next;

}

ptr->next = NULL;

free(temp);

}

return 0;

}

int deleteloc(int loc)

{

struct node\* temp= head;

struct node\* ptr;

for(int i=1; i<loc; i++)

{

ptr = temp;

temp = temp->next;

}

ptr->next = temp->next;

temp->next->prev = ptr;

free(temp);

return 0;

}

void traverse()

{

struct node\* temp = head;

while(temp->next!=NULL)

{

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

temp = temp->next;

}

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

}

void main()

{

int ch=1;

while(ch==1)

{

int choice,d;

printf("Enter 1 for Insertbeg, 2 for Insertend, 3 for Insertloc\n");

printf("Enter 4 for deletebeg, 5 for Deleteend, 6 for Deleteloc\n");

printf("Enter 7 for traversal\n");

scanf("%d",&choice);

switch (choice)

{

case 1:

{

printf("Input Data\n");

scanf("%d",&d);

insertbeg(d);

break;

}

case 2:

{

printf("Input Data\n");

scanf("%d",&d);

insertend(d);

break;

}

case 3:

{

int l;

printf("Enter The location\n");

scanf("%d",&l);

printf("Input Data\n");

scanf("%d",&d);

insertloc(l,d);

break;

}

case 4:

{

deletebeg();

break;

}

case 5:

{

deleteend();

break;

}

case 6:

{

int loc;

printf("Enter The location\n");

scanf("%d",&loc);

deleteloc(loc);

break;

}

case 7:

{

traverse();

}

}

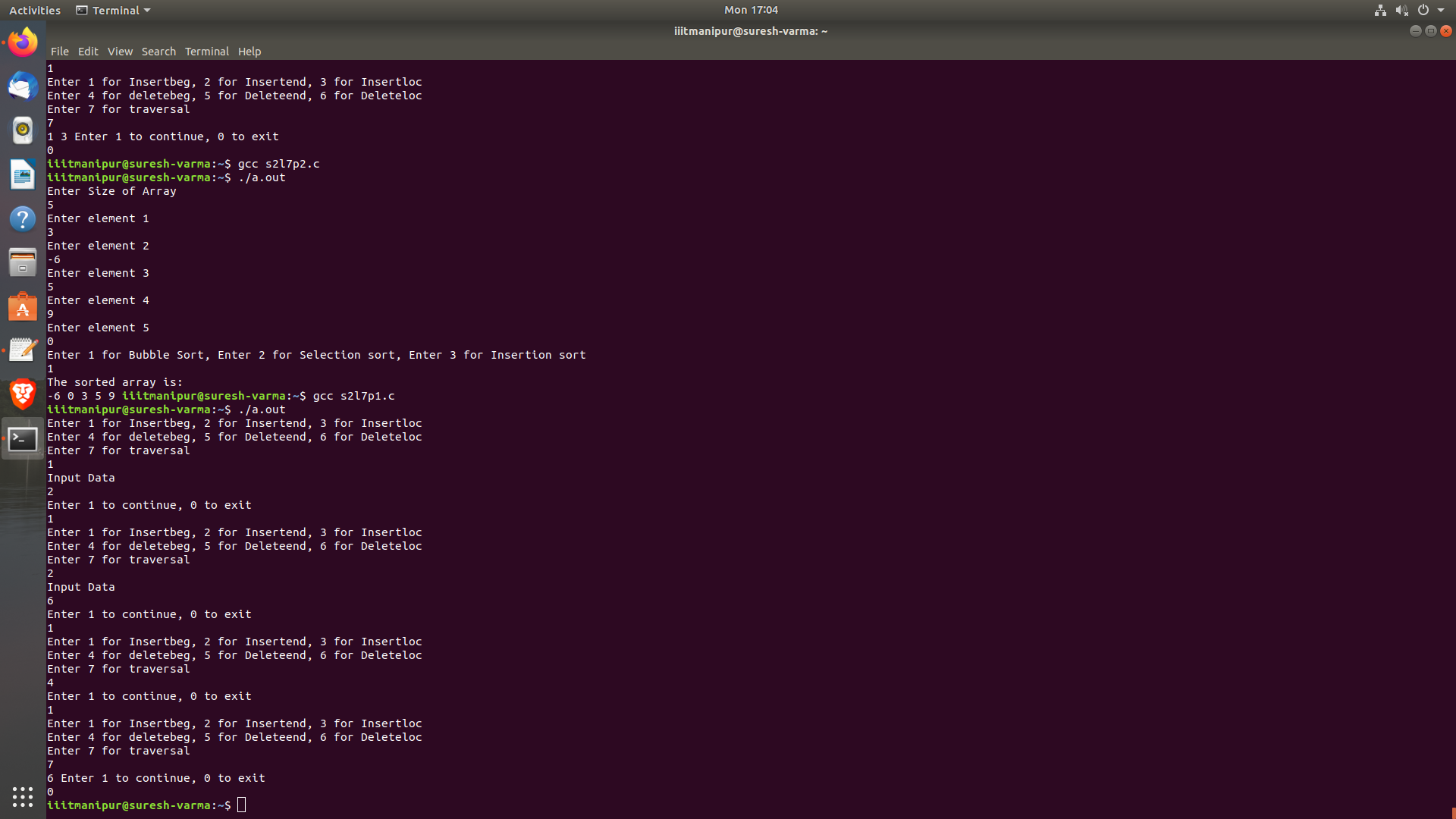
printf("Enter 1 to continue, 0 to exit\n");

scanf("%d",&ch);

}

}

**OUTPUT:**

****