Institute of Computer Technology

B. Tech Computer Science and Engineering

Subject: DS (2CSE302)

**PRACTICAL-18**

**AIM: - Implement the scenario based on doubly linked list & circular doubly linked list.**

***Ques. Jay wants to add forward and backward button in his website which directs the addresses to previous and next from the current value. Refer the given scenario and provide appropriate solution using doubly linked list:***

**1. Insert new item at the first position**

**2. Insert new item at the last position**

**3. Insert new item before the given value of node**

**4. Insert new item after the given value of node**

**5. Delete item from the first position**

**6. Delete item from the last position**

**7. Delete particular item**

**8. Delete item before the given item**

**9. Delete item after the given item**

**10. Search item from the list**

**11. Display all elements of the list**

***SOLUTION***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*previous, \*next;

};

struct node \*start;

void insert\_begin()

{

int yash;

printf("Enter element you want to insert at first position: ");

scanf("%d", &yash);

struct node \*new\_node;

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = yash;

if (start == NULL)

{

start = new\_node;

new\_node->next = NULL;

new\_node->previous = NULL;

}

else

{

start->previous = new\_node;

new\_node->next = start;

new\_node->previous = NULL;

start = new\_node;

}

}

void insert\_end()

{

int yash;

printf("Enter element you want to insert at last position: ");

scanf("%d", &yash);

struct node \*new\_node, \*ptr;

new\_node = (struct node \*)malloc(sizeof(struct node));

ptr = start;

new\_node->data = yash;

new\_node->next = NULL;

if (start == NULL)

{

start = new\_node;

new\_node->previous = NULL;

}

else

{

while (ptr->next != NULL)

{

ptr = ptr->next;

}

new\_node->previous = ptr;

ptr->next = new\_node;

}

}

void insert\_before()

{

int yash, item;

printf("Enter element you want to insert: ");

scanf("%d", &yash);

printf("Enter element before which yo want to insert: ");

scanf("%d", &item);

struct node \*new\_node, \*ptr, \*before;

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = yash;

ptr = start;

before = ptr;

while (ptr->data != item)

{

before = ptr;

ptr = ptr->next;

}

before->next = new\_node;

new\_node->previous = before;

new\_node->next = ptr;

ptr->previous = new\_node;

}

void insert\_after()

{

int yash, item;

printf("Enter element you want to insert: ");

scanf("%d", &yash);

printf("Enter element after which yo want to insert: ");

scanf("%d", &item);

struct node \*new\_node, \*ptr;

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = yash;

ptr = start;

while (ptr->data != item)

{

ptr = ptr->next;

}

new\_node->next = ptr->next;

new\_node->previous = ptr;

ptr->next = new\_node;

}

void delete\_begin()

{

struct node \*ptr;

ptr = start;

start = start->next;

start->previous = NULL;

free(ptr);

}

void delete\_last()

{

struct node \*ptr;

ptr = start;

while (ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->previous->next = NULL;

free(ptr);

}

void delete ()

{

int yash;

printf("Enter element which you want to delete: ");

scanf("%d", &yash);

struct node \*ptr;

ptr = start;

while (ptr->data != yash)

{

ptr = ptr->next;

}

ptr->previous->next = ptr->next;

ptr->next->previous = ptr->previous;

free(ptr);

}

void delete\_before()

{

int yash;

printf("Enter element before which you want to delete: ");

scanf("%d", &yash);

struct node \*ptr, \*before;

ptr = start;

before = ptr;

while (ptr->data != yash)

{

before = ptr;

ptr = ptr->next;

}

ptr->previous = ptr->previous->previous;

before->previous->next = ptr;

free(before);

}

void delete\_after()

{

int yash;

printf("Enter element after which you want to delete: ");

scanf("%d", &yash);

struct node \*ptr, \*after;

ptr = start;

while (ptr->data != yash)

{

ptr = ptr->next;

}

after = ptr->next;

ptr->next = after->next;

after->next->previous = after->previous;

free(after);

}

void search()

{

int yash, pos = 0;

printf("Insert item which you want to search: ");

scanf("%d", &yash);

struct node \*ptr;

ptr = start;

while (ptr->next != NULL)

{

if (ptr->data == yash)

{

break;

}

else

{

ptr = ptr->next;

pos++;

}

}

printf("Item is available at the %d position of the list\n", pos + 1);

}

void display()

{

printf("List is: \n");

struct node \*ptr;

ptr = start;

while (ptr != NULL)

{

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

ptr = ptr->next;

}

}

int main()

{

int choice;

for (;;)

{

printf("\n1. Insert new item at the first position\n2. Insert new item at the last position\n3. Insert new item before the given value of node\n4. Insert new item after the given value of node\n5. Delete item from the first position\n6. Delete item from the last position\n7. Delete particular item\n8. Delete item before the given item\n9. Delete item after the given item\n10. Search item from the list\n11. Display all elements of the list\n12. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert\_begin();

break;

case 2:

insert\_end();

break;

case 3:

insert\_before();

break;

case 4:

insert\_after();

break;

case 5:

delete\_begin();

break;

case 6:

delete\_last();

break;

case 7:

delete ();

break;

case 8:

delete\_before();

break;

case 9:

delete\_after();

break;

case 10:

search();

break;

case 11:

display();

break;

case 12:

exit(0);

break;

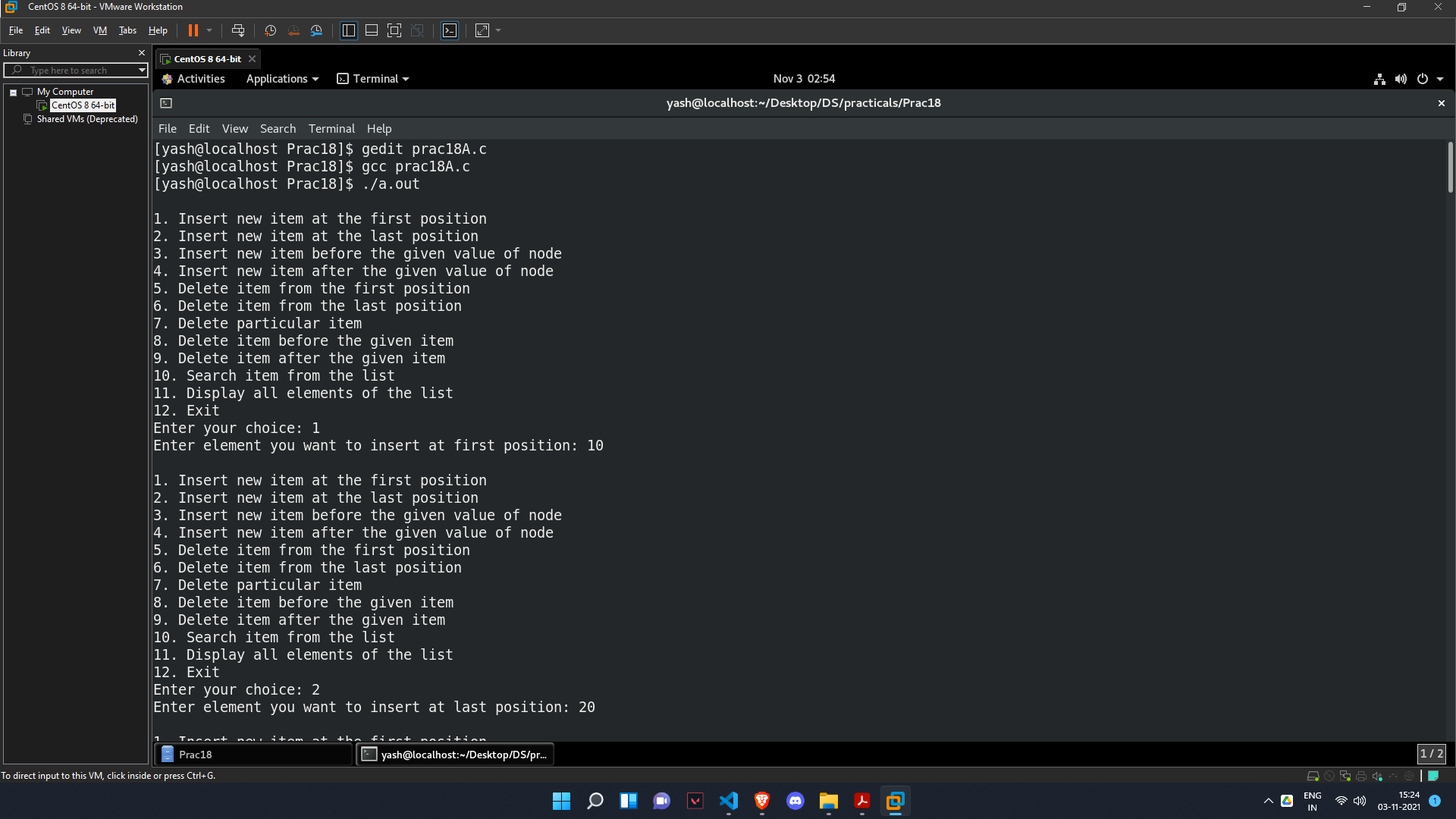
}

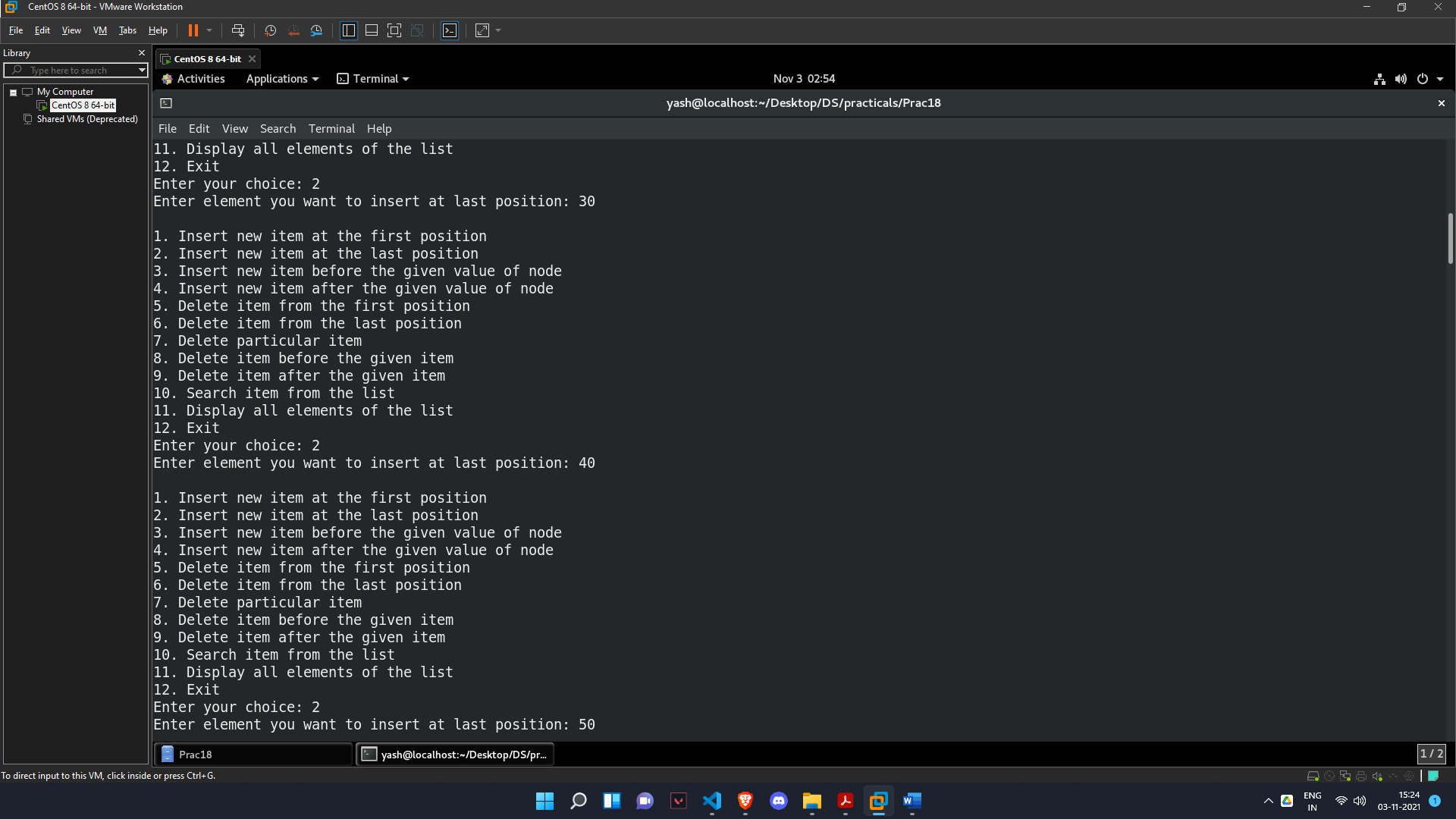
}

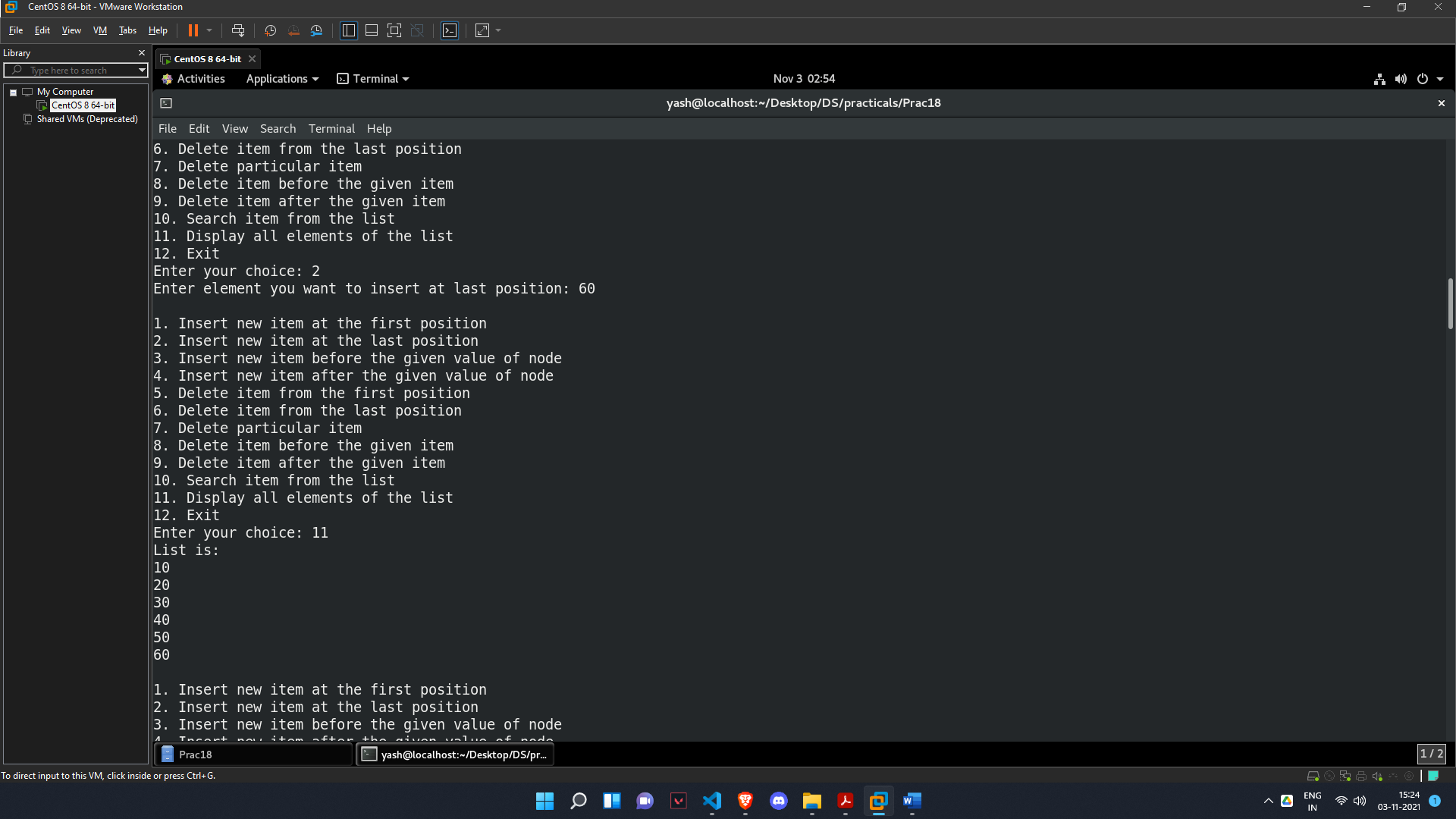
return 0;

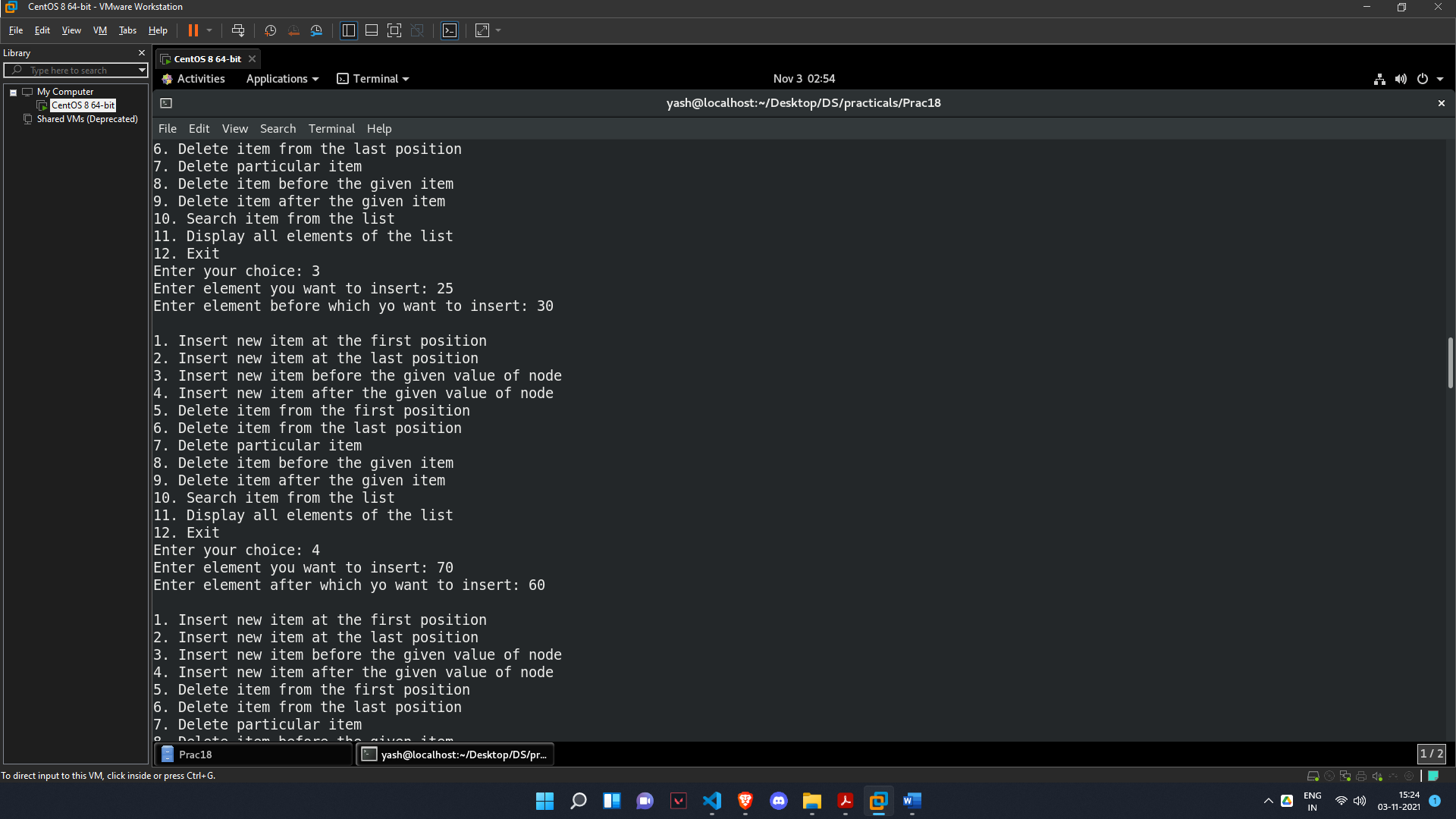
}

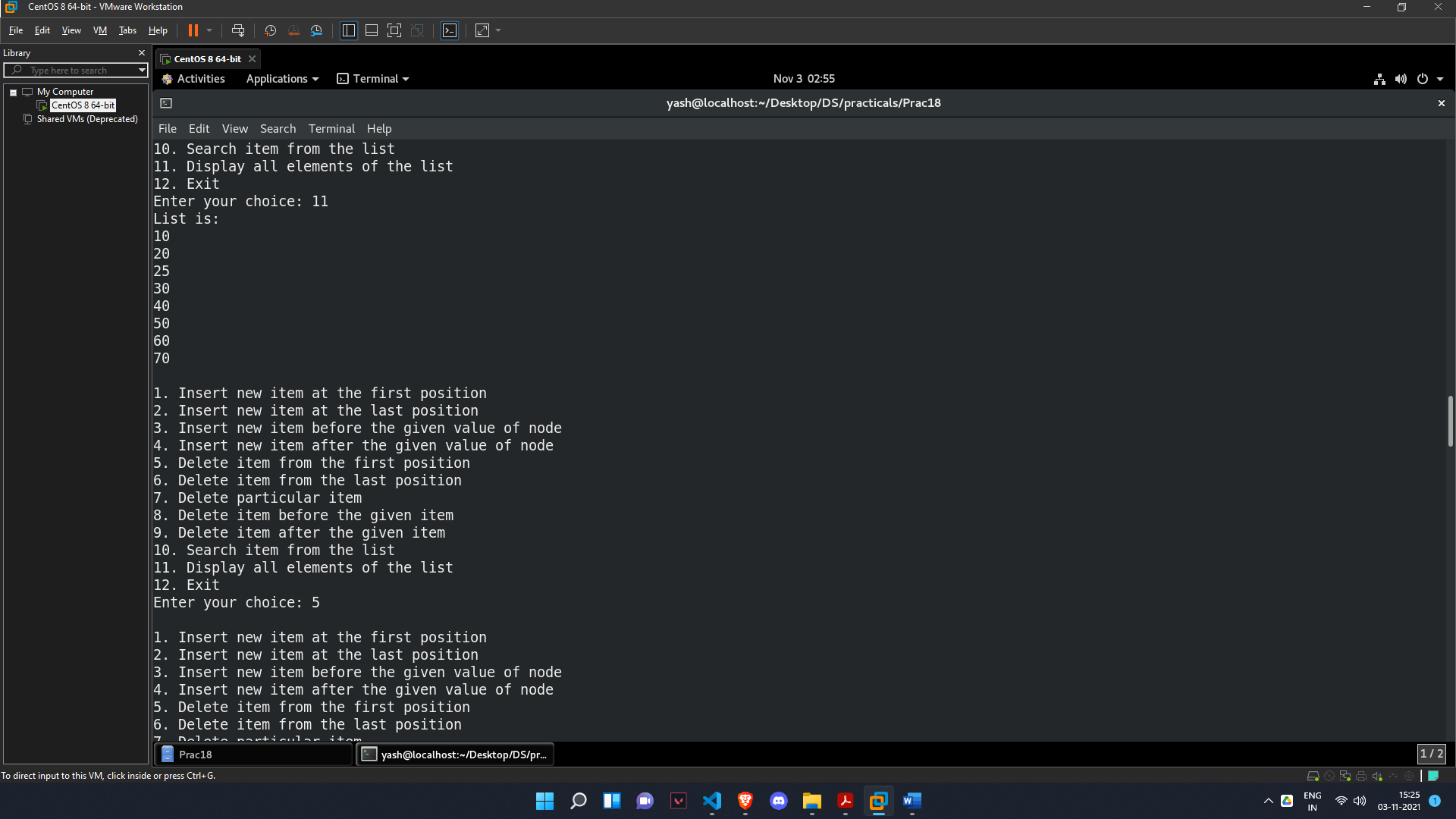
***OUTPUT***

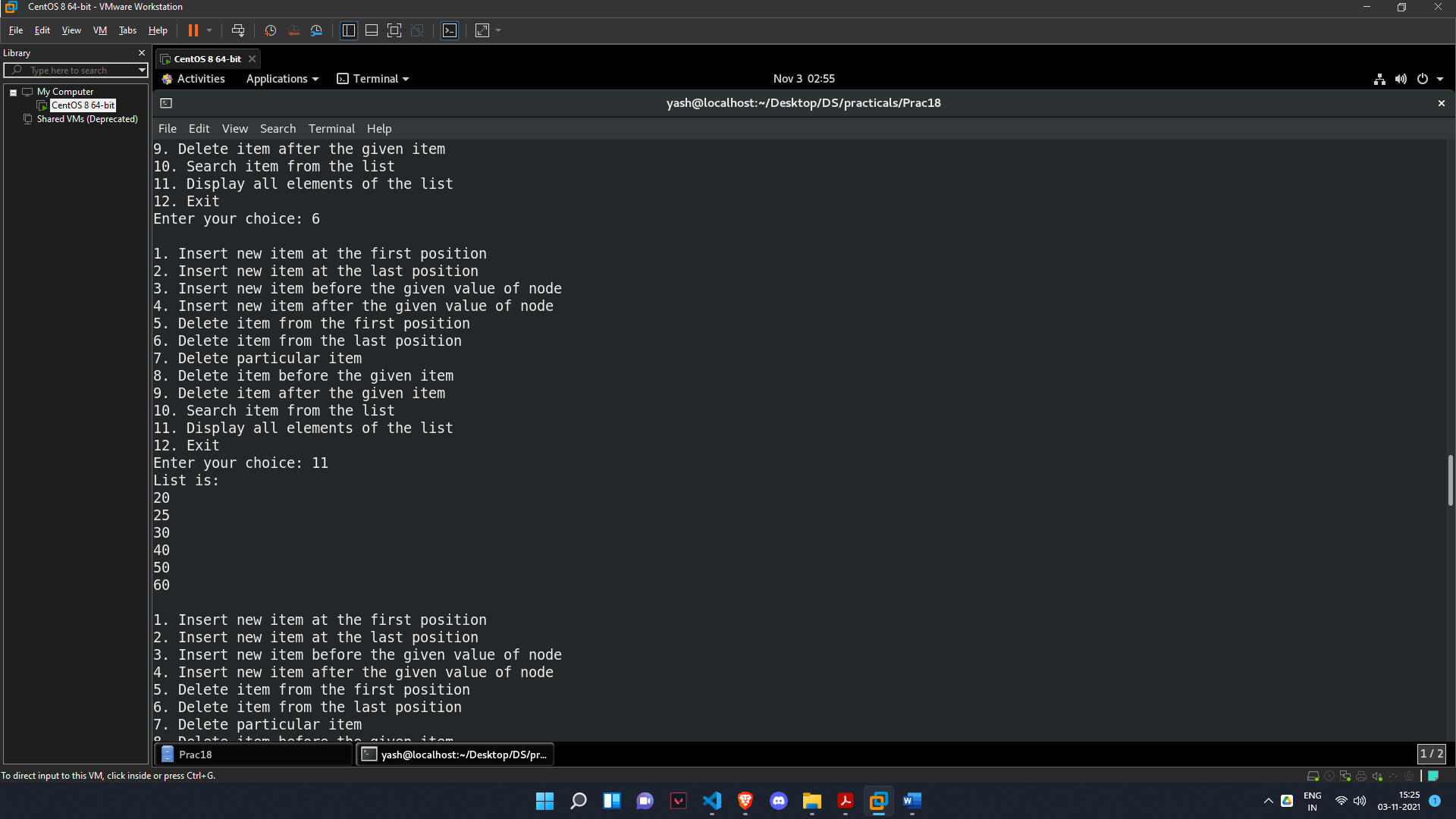


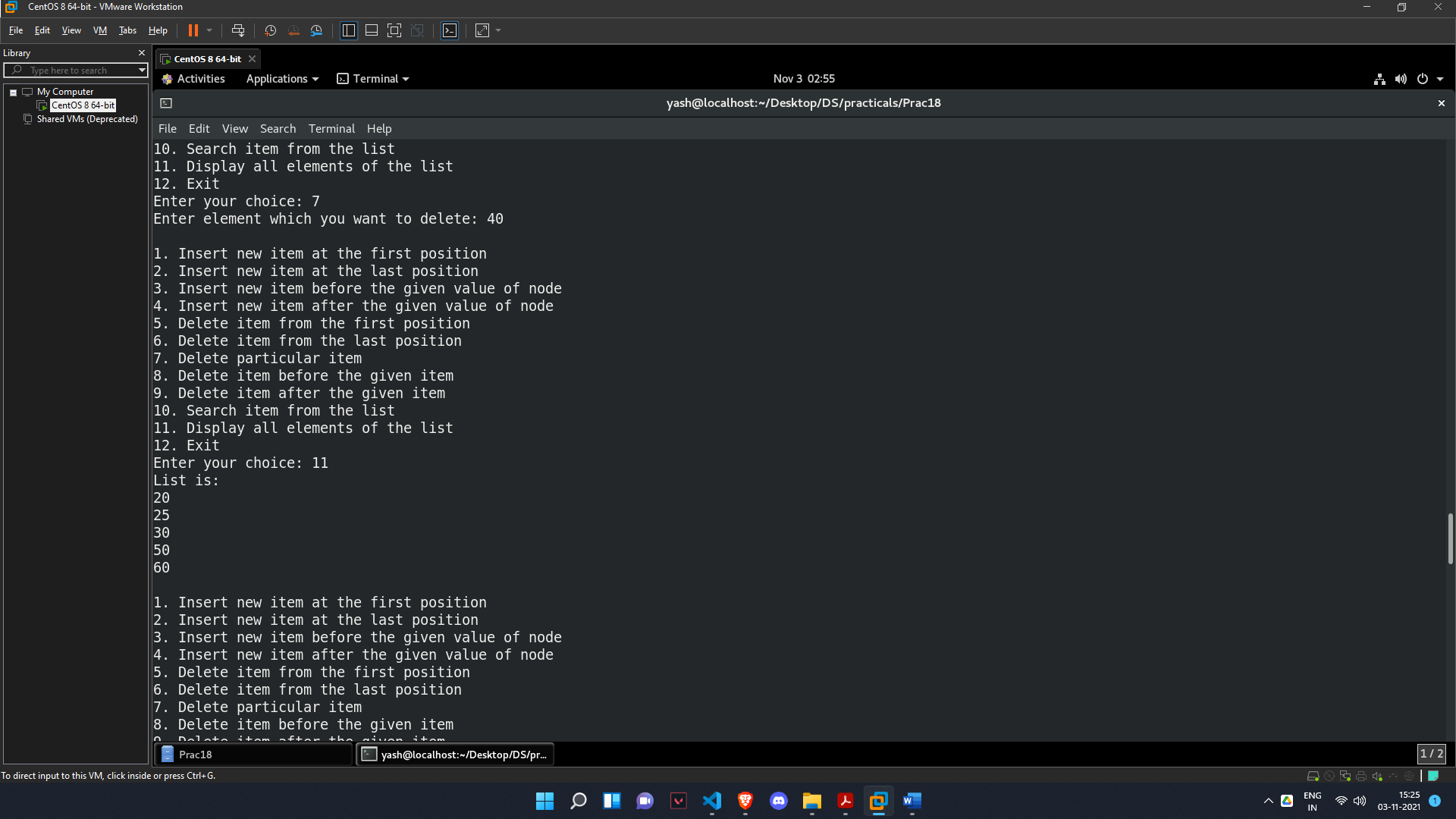


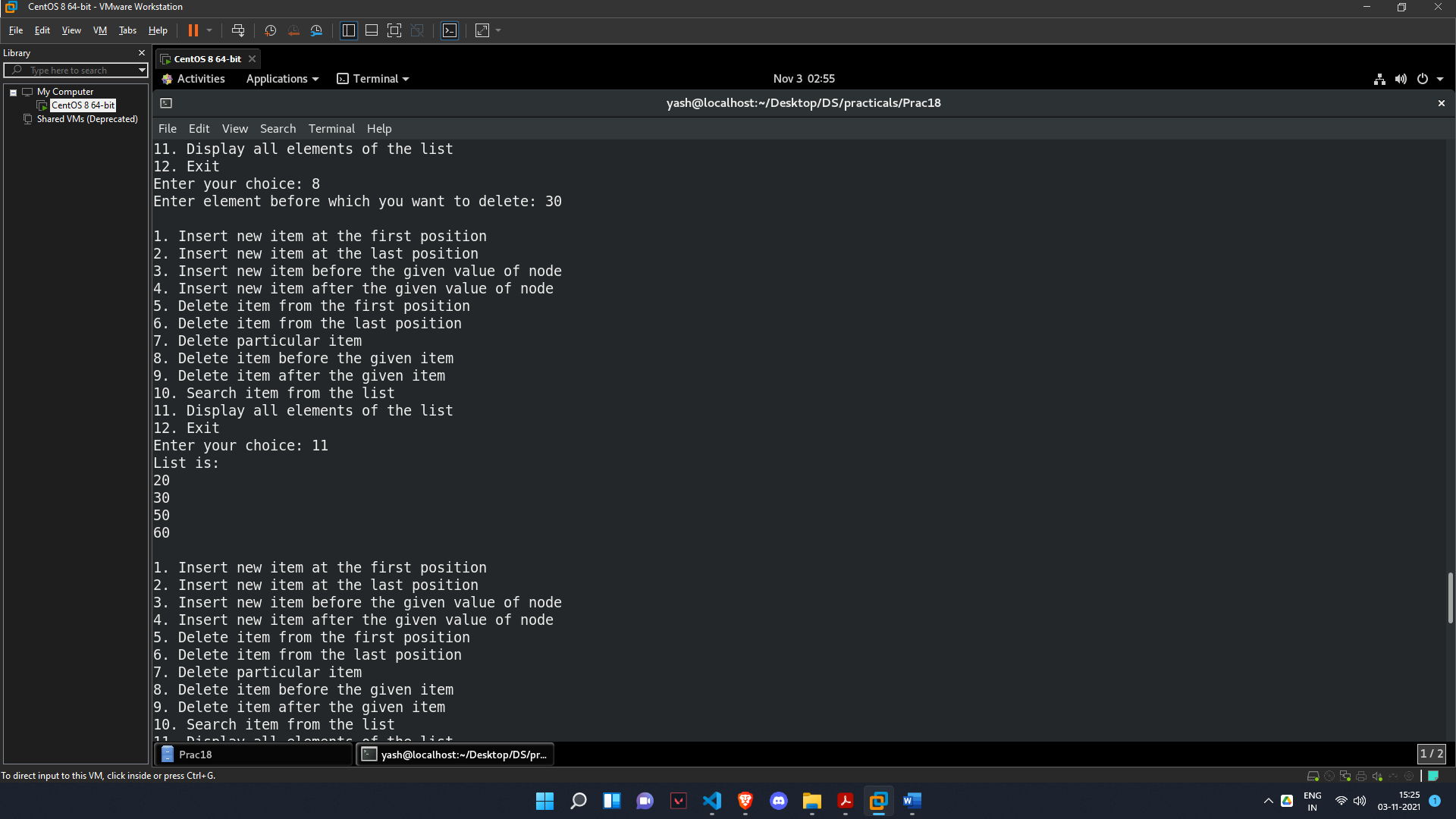


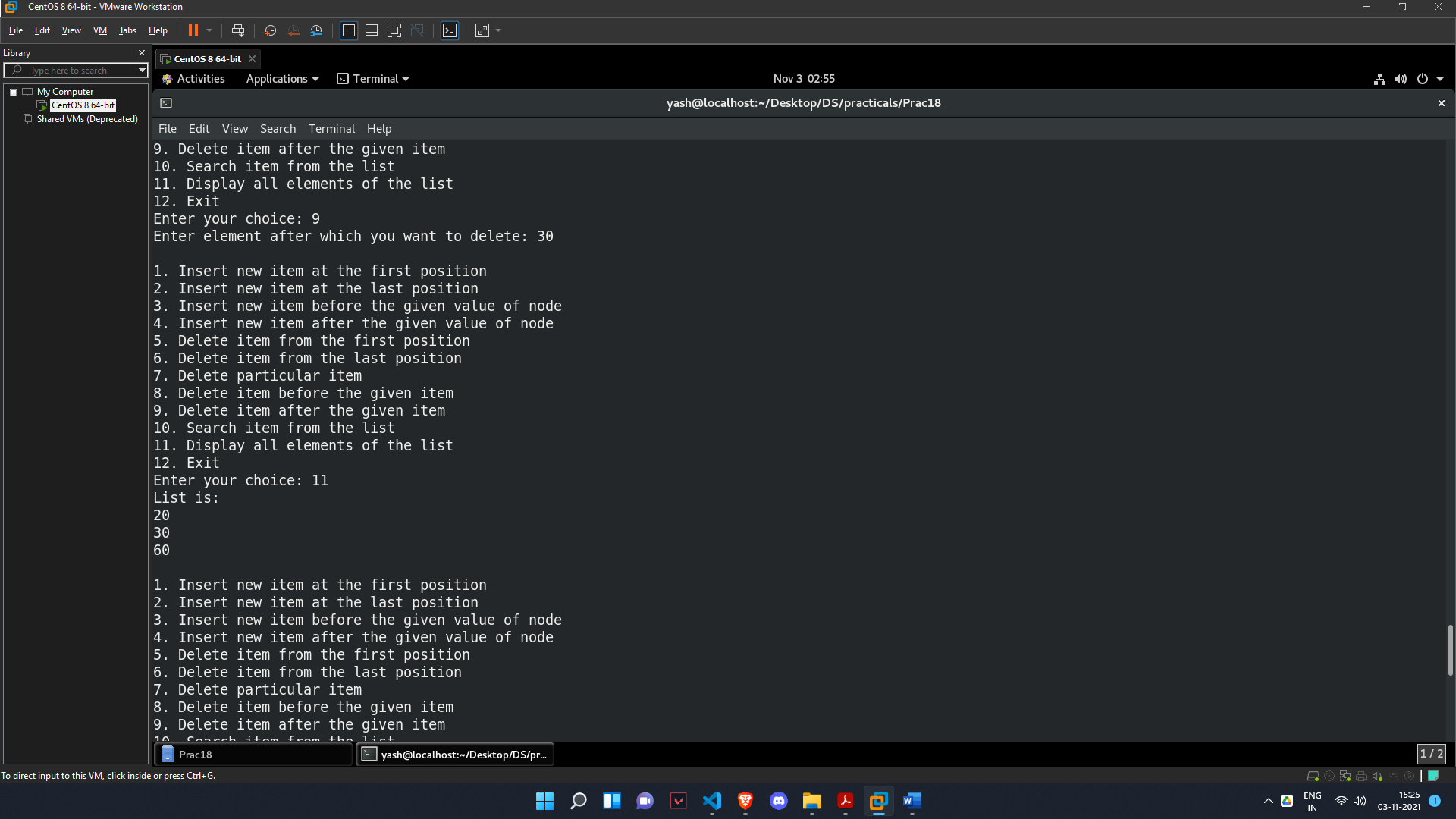


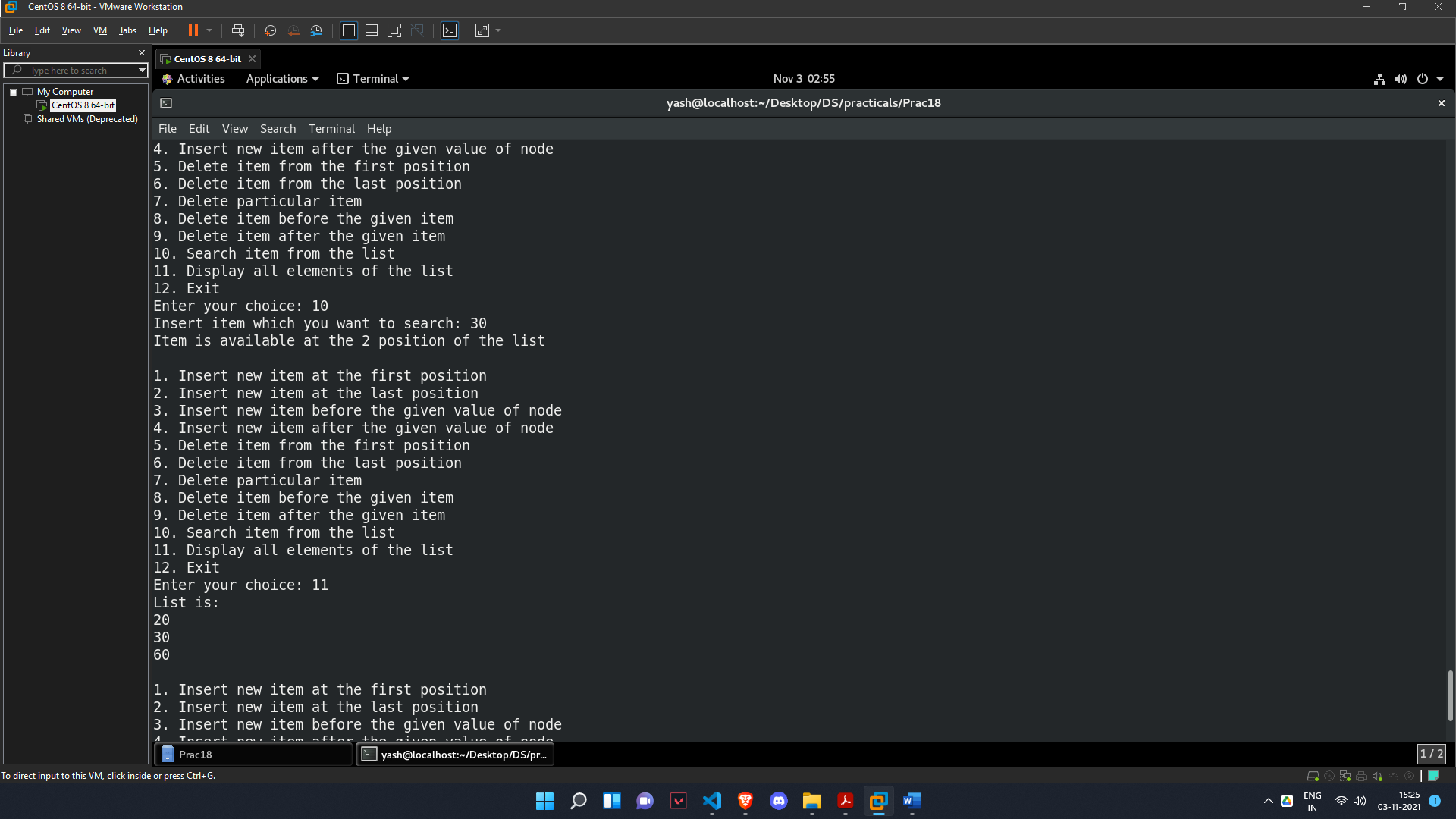


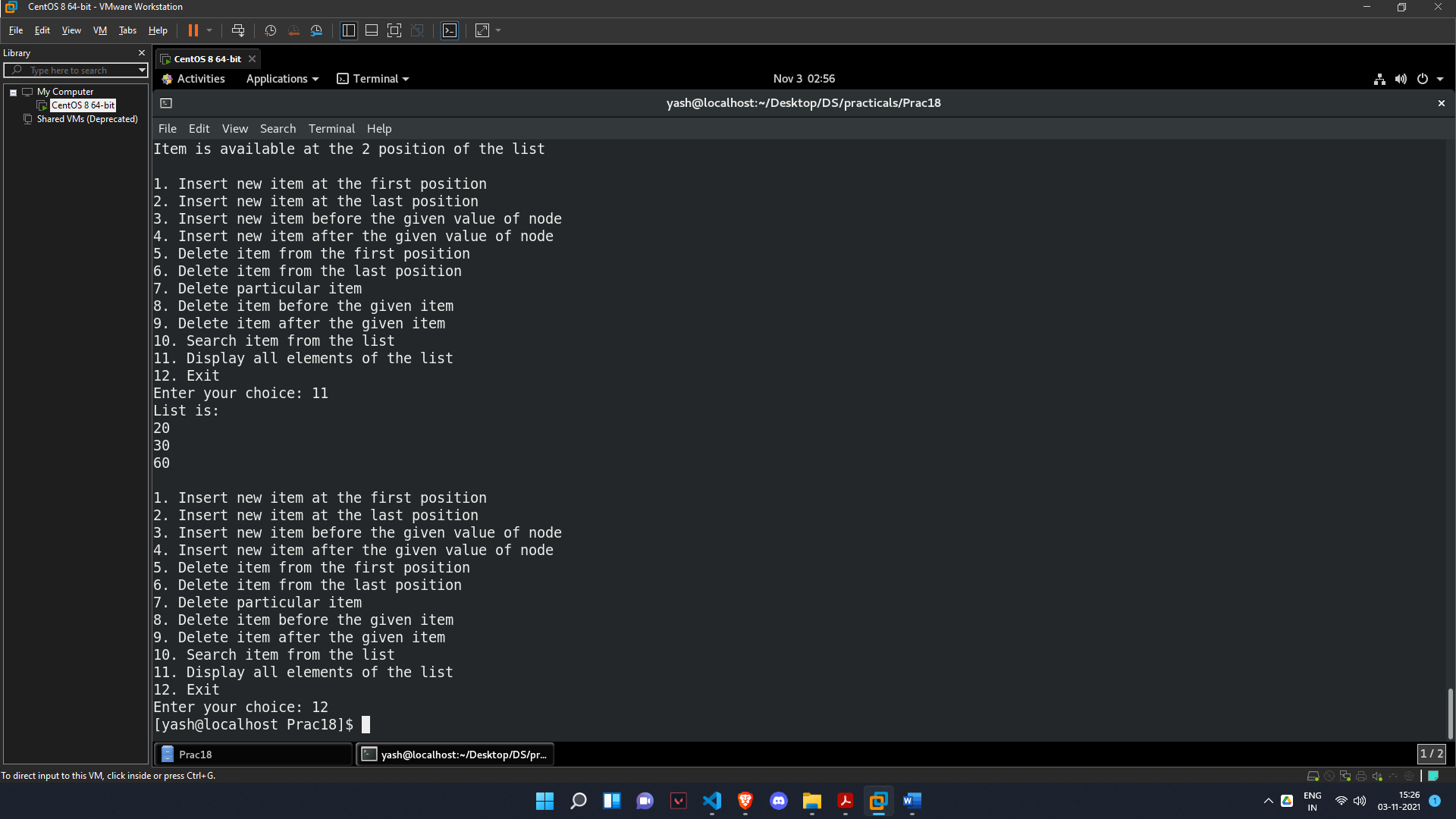












**Homework:**

***Ques. Jay wants to add forward and backward button in his website which directs the addresses to previous and next from the current value. Refer the given scenario and provide appropriate solution using circular doubly linked list:***

**1. Insert new item at the first position**

**2. Insert new item at the last position**

**3. Insert new item before the given value of node**

**4. Insert new item after the given value of node**

**5. Delete item from the first position**

**6. Delete item from the last position**

**7. Delete particular item**

**8. Delete item before the given item**

**9. Delete item after the given item**

**10. Search item from the list**

**11. Display all elements of the list**

***SOLUTION***

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*previous, \*next;

};

struct node \*start;

void insert\_begin()

{

int yash;

printf("Enter element you want to insert at first position: ");

scanf("%d", &yash);

struct node \*new\_node, \*ptr;

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = yash;

ptr = start;

if (start == NULL)

{

start = new\_node;

new\_node->next = start;

new\_node->previous = start;

}

else

{

start->previous = new\_node;

new\_node->next = start;

while (ptr->next != start)

{

ptr = ptr->next;

}

new\_node->previous = ptr;

start = new\_node;

}

}

void insert\_end()

{

int yash;

printf("Enter element you want to insert at last position: ");

scanf("%d", &yash);

struct node \*new\_node, \*ptr;

new\_node = (struct node \*)malloc(sizeof(struct node));

ptr = start;

new\_node->data = yash;

new\_node->next = start;

if (start == NULL)

{

start = new\_node;

new\_node->previous = start;

}

else

{

while (ptr->next != start)

{

ptr = ptr->next;

}

new\_node->previous = ptr;

ptr->next = new\_node;

}

}

void insert\_before()

{

int yash, item;

printf("Enter element you want to insert: ");

scanf("%d", &yash);

printf("Enter element before which yo want to insert: ");

scanf("%d", &item);

struct node \*new\_node, \*ptr, \*before;

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = yash;

ptr = start;

before = ptr;

while (ptr->data != item)

{

before = ptr;

ptr = ptr->next;

}

before->next = new\_node;

new\_node->previous = before;

new\_node->next = ptr;

ptr->previous = new\_node;

}

void insert\_after()

{

int yash, item;

printf("Enter element you want to insert: ");

scanf("%d", &yash);

printf("Enter element after which yo want to insert: ");

scanf("%d", &item);

struct node \*new\_node, \*ptr;

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = yash;

ptr = start;

while (ptr->data != item)

{

ptr = ptr->next;

}

new\_node->next = ptr->next;

new\_node->previous = ptr;

ptr->next = new\_node;

}

void delete\_begin()

{

struct node \*ptr;

ptr = start;

while (ptr->next != start)

{

ptr = ptr->next;

}

ptr->next = start->next;

free(start);

start = ptr->next;

ptr->next->previous = ptr;

}

void delete\_last()

{

struct node \*ptr;

ptr = start;

while (ptr->next != start)

{

ptr = ptr->next;

}

ptr->previous->next = start;

free(ptr);

}

void delete ()

{

int yash;

printf("Enter element which you want to delete: ");

scanf("%d", &yash);

struct node \*ptr;

ptr = start;

while (ptr->data != yash)

{

ptr = ptr->next;

}

ptr->previous->next = ptr->next;

ptr->next->previous = ptr->previous;

free(ptr);

}

void delete\_before()

{

int yash;

printf("Enter element before which you want to delete: ");

scanf("%d", &yash);

struct node \*ptr, \*before;

ptr = start;

before = ptr;

while (ptr->data != yash)

{

before = ptr;

ptr = ptr->next;

}

ptr->previous = ptr->previous->previous;

before->previous->next = ptr;

free(before);

}

void delete\_after()

{

int yash;

printf("Enter element after which you want to delete: ");

scanf("%d", &yash);

struct node \*ptr, \*after;

ptr = start;

while (ptr->data != yash)

{

ptr = ptr->next;

}

after = ptr->next;

ptr->next = after->next;

after->next->previous = after->previous;

free(after);

}

void search()

{

int yash, pos = 0;

printf("Insert item which you want to search: ");

scanf("%d", &yash);

struct node \*ptr;

ptr = start;

while (ptr->next != start)

{

if (ptr->data == yash)

{

break;

}

else

{

ptr = ptr->next;

pos++;

}

}

printf("Item is available at the %d position of the list\n", pos + 1);

}

void display()

{

printf("List is:\n");

struct node \*ptr;

ptr = start;

while (ptr->next != start)

{

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

ptr = ptr->next;

}

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

}

int main()

{

int choice;

for (;;)

{

printf("\n1. Insert new item at the first position\n2. Insert new item at the last position\n3. Insert new item before the given value of node\n4. Insert new item after the given value of node\n5. Delete item from the first position\n6. Delete item from the last position\n7. Delete particular item\n8. Delete item before the given item\n9. Delete item after the given item\n10. Search item from the list\n11. Display all elements of the list\n12. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert\_begin();

break;

case 2:

insert\_end();

break;

case 3:

insert\_before();

break;

case 4:

insert\_after();

break;

case 5:

delete\_begin();

break;

case 6:

delete\_last();

break;

case 7:

delete ();

break;

case 8:

delete\_before();

break;

case 9:

delete\_after();

break;

case 10:

search();

break;

case 11:

display();

break;

case 12:

exit(0);

break;

}

}

return 0;

}

***OUTPUT***

