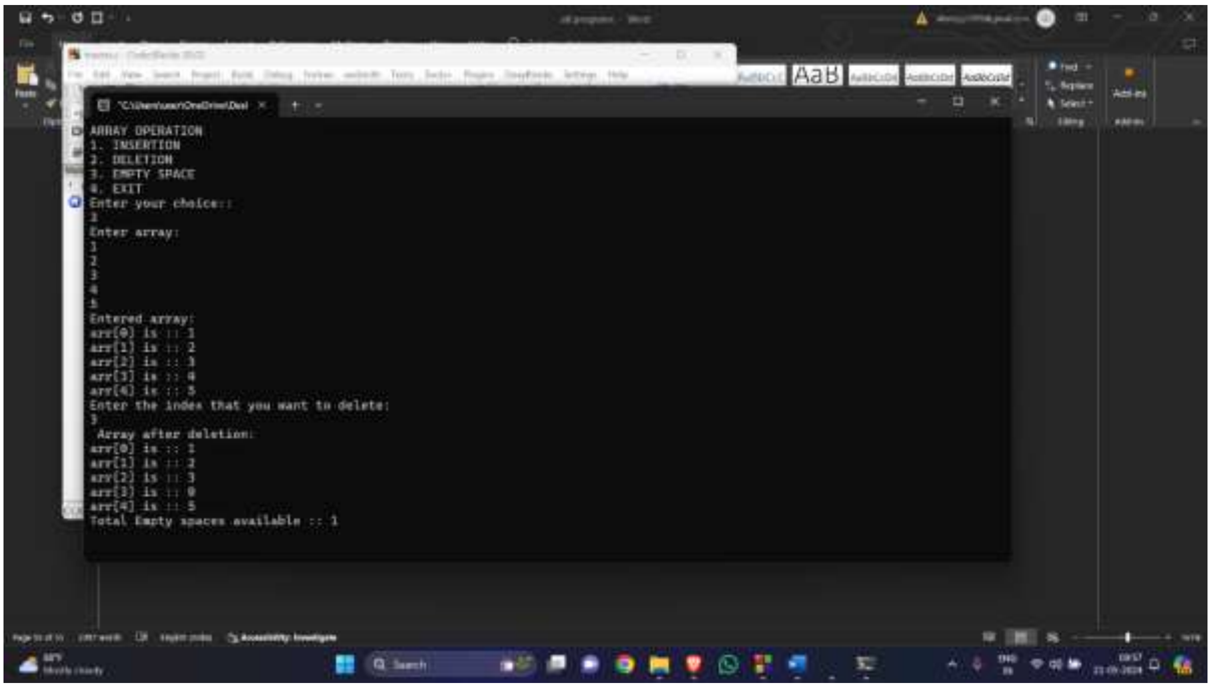
DAY 6

1. Array operation

```
#include<stdio.h>
void main()
{
  int choice;
    start:
  printf("ARRAY OPERATION\n");
  printf("1. INSERTION\n");
  printf("2. DELETION\n");
  printf("3. EMPTY SPACE\n");
  printf("4. EXIT\n");
  printf("Enter your choice::\n");
  scanf("%d",&choice);
  switch(choice)
  {
  case 1:
      {
        insert();
        int y;
        printf("\nDo you want to continue?(0/1)?\n");
        scanf("%d",&y);
         if(y==1){
           system("cls");
           goto start;
        }
        else
        {
        printf("Bye....");
        }
        break;
      }
```

```
case 2:
    {
        delete();
        int y;
        printf("\nDo you want to continue?(0/1)?\n");
        scanf("%d",&y);
         if(y==1){
            system("cls");
            goto start;
        }
        else
        {
        printf("Bye....");
        }
        break;
    }
case 3:
    {
        empty();
        int y;
        printf("\nDo you want to continue?(0/1)?\n");
        scanf("%d",&y);
         if(y==1){
            system("cls");
            goto start;
        }
        else
        {
        printf("Bye....");
        }
        break;
    }
    case 4:
        {
            printf("Bye....");
            break;
        }
    default:
        {
             printf("Bye....");
            break;
        }
    }
}
void insert()
{
    int arr[5];
    int i;
    printf("Enter array:\n");
    for(i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
   printf("Entered array:\n");
    for(i=0;i<5;i++)
    {
        printf("arr[%d] is :: %d\n",i,arr[i]);
    }
    getch();
}
void delete()
{
    int n,i,arr[5];
    printf("Enter array:\n");
    for(i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
   printf("Entered array:\n");
    for(i=0;i<5;i++)
    {
        printf("arr[%d] is :: %d\n",i,arr[i]);
    }
    printf("Enter the index that you want to delete:\n");
    scanf("%d",&n);
    for(i=0;i<=n;i++)
    {
        if(i==n)
        {
            arr[i]=0;
        }
    }
    printf(" Array after deletion:\n");
    for(int j=0;j<5;j++)
    {
        printf("arr[%d] is :: %d\n",j,arr[j]);
    }
    getch();
}
void empty()
{

     int counter=0,i,arr[5],n;
     printf("Enter array:\n");
    for(i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
   printf("Entered array:\n");
    for(i=0;i<5;i++)
    {
        printf("arr[%d] is :: %d\n",i,arr[i]);
    }
    printf("Enter the index that you want to delete:\n");
    scanf("%d",&n);
    for(i=0;i<=n;i++)
    {
        if(i==n)
        {
            arr[i]=0;
        }
    }
    printf(" Array after deletion:\n");
    for(int j=0;j<5;j++)
    {
        printf("arr[%d] is :: %d\n",j,arr[j]);
    }
    for (i=0;i<5;i++)
    {
        if (arr[i]== 0)
        counter = counter +1;
    }
    printf("Total Empty spaces available :: %d\n", counter);
getch();
}
```

OUTPUT:



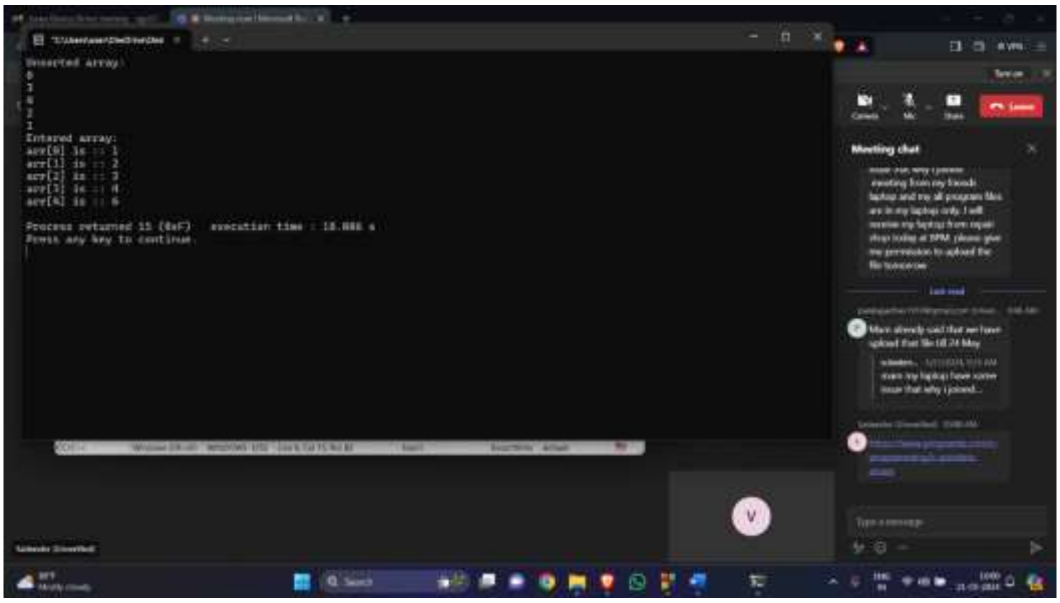
```
ARRAY OPERATION
1. INSERTION
2. DELETION
3. EMPTY SPACE
4. EXIT
Enter your choice::
2
Enter array:
1
2
3
4
5
Entered array:
arr[0] is :: 1
arr[1] is :: 2
arr[2] is :: 3
arr[3] is :: 4
arr[4] is :: 5
Enter the index that you want to delete:
3
 Array after deletion:
arr[0] is :: 1
arr[1] is :: 2
arr[2] is :: 3
arr[3] is :: 0
arr[4] is :: 5
Total Empty spaces available :: 1
```

2. Binary search

```
#include<stdio.h>
void main()
{
   int arr[5];
   int i,n=5;
   printf("Unsorted array:\n");
   for(i=0;i<5;i++){
scanf("%d",&arr[i]);
   }
  int j,temp;
   for (i = 0; i < n-1; i++) {
      for (j = 0; j < n-i-1; j++) {
         if (*(arr + j) > *(arr + j + 1)) {
            temp = *(arr + j);
            *(arr + j) = *(arr + j + 1);
            *(arr + j + 1) = temp;
         }
      }
```

```
    }
    printf("Entered array:\n");
    for(i=0;i<5;i++)
    {
        printf("arr[%d] is :: %d\n",i,arr[i]);
    }
}
```

OUTPUT:

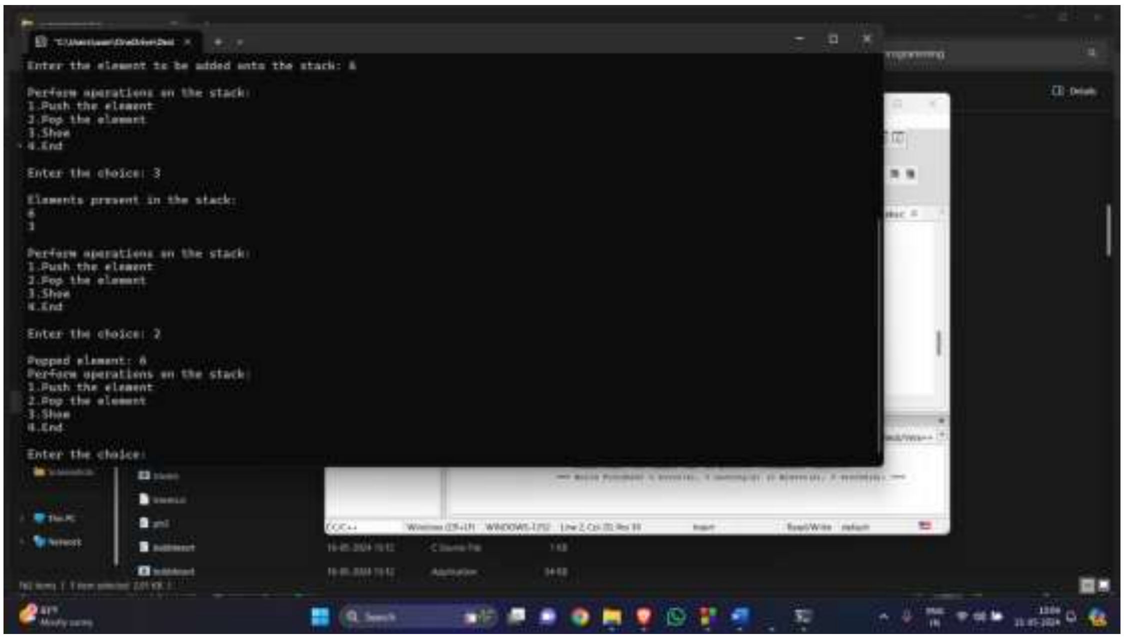


```
Unsorted array:
6
3
4
2
1
Entered array:
arr[0] is :: 1
arr[1] is :: 2
arr[2] is :: 3
arr[3] is :: 4
arr[4] is :: 6

Process returned 15 (0xF)   execution time : 18.888 s
Press any key to continue.
```

3. Stack operation

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4

int top = -1, inp_array[SIZE];
void push();
void pop();
void show();

int main()
{
    int choice;
    while (1)
    {
        printf("\nPerform operations on the stack:");
        printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");
        printf("\n\nEnter the choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            show();
            break;
        case 4:
            exit(0);

        default:
            printf("\nInvalid choice!!");
        }
    }
}

void push()
{
    int x;
    if (top == SIZE - 1)
    {
        printf("\nOverflow!!");
    }
    else
    {
        printf("\nEnter the element to be added onto the stack: ");
        scanf("%d", &x);
        top = top + 1;
        inp_array[top] = x;
    }
}

void pop()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nPopped element: %d", inp_array[top]);
        top = top - 1;
    }
}

void show()
{
    if (top == -1)
    {
        printf("\nUnderflow!!");
    }
    else
    {
        printf("\nElements present in the stack: \n");
        for (int i = top; i >= 0; --i)
            printf("%d\n", inp_array[i]);
    }
}
```

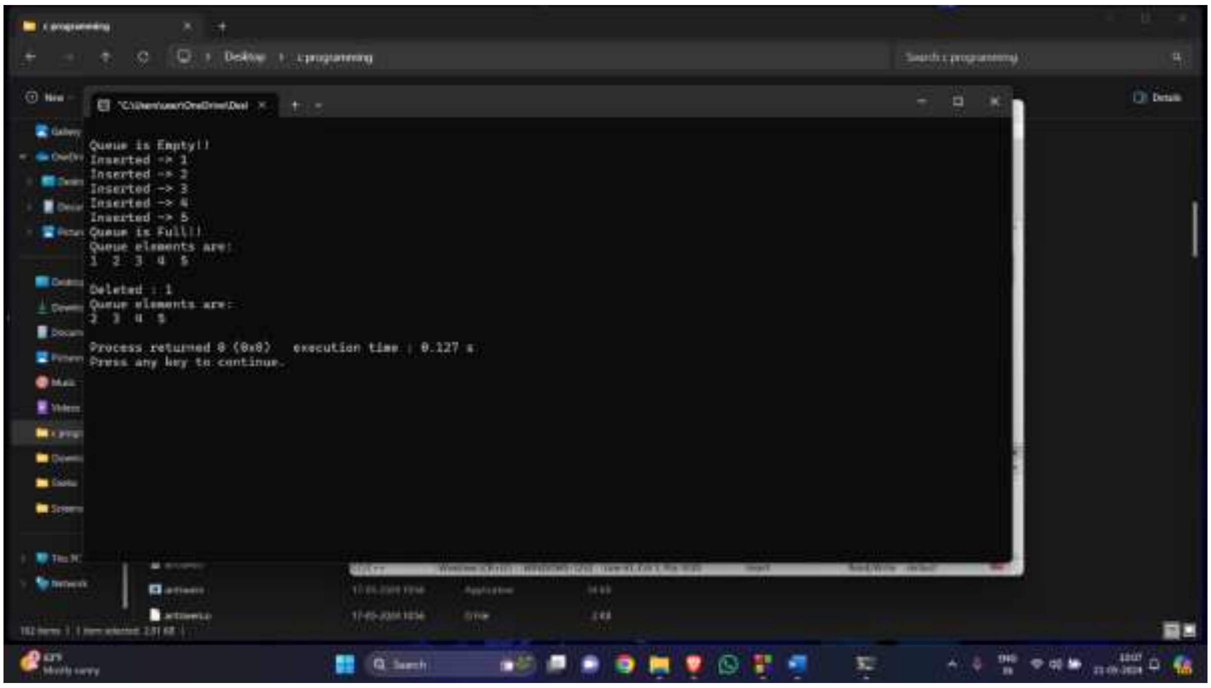
OUTPUT:



```
Enter the element to be added onto the stack: 6

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 3

Elements present in the stack:
6
3

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 2

Popped element: 6
Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice:
```

4. Queue operation

```
// Queue implementation in C


#include <stdio.h>
#define SIZE 5


void enQueue(int);
void deQueue();
void display();


int items[SIZE], front = -1, rear = -1;


int main() {
  //deQueue is not possible on empty queue
  deQueue();

  //enQueue 5 elements
  enQueue(1);
  enQueue(2);
  enQueue(3);
  enQueue(4);
  enQueue(5);

  // 6th element can't be added to because the queue is full
  enQueue(6);

  display();

  //deQueue removes element entered first i.e. 1
  deQueue();

  //Now we have just 4 elements
  display();

  return 0;
}

void enQueue(int value) {
  if (rear == SIZE - 1)
    printf("\nQueue is Full!!");
  else {
    if (front == -1)
      front = 0;
    rear++;
    items[rear] = value;
    printf("\nInserted -> %d", value);
  }
}

void deQueue() {
  if (front == -1)
    printf("\nQueue is Empty!!");
  else {
    printf("\nDeleted : %d", items[front]);
    front++;
    if (front > rear)
      front = rear = -1;
  }
}

// Function to print the queue
void display() {
  if (rear == -1)
    printf("\nQueue is Empty!!!");
  else {
    int i;
    printf("\nQueue elements are:\n");
    for (i = front; i <= rear; i++)
      printf("%d  ", items[i]);
  }
  printf("\n");
}
```
OUTPUT:



```
Queue is Empty!!
Inserted -> 1
Inserted -> 2
Inserted -> 3
Inserted -> 4
Inserted -> 5
Queue is Full!!
Queue elements are:
1  2  3  4  5

Deleted : 1
Queue elements are:
2  3  4  5

Process returned 0 (0x0)   execution time : 0.127 s
Press any key to continue.
```

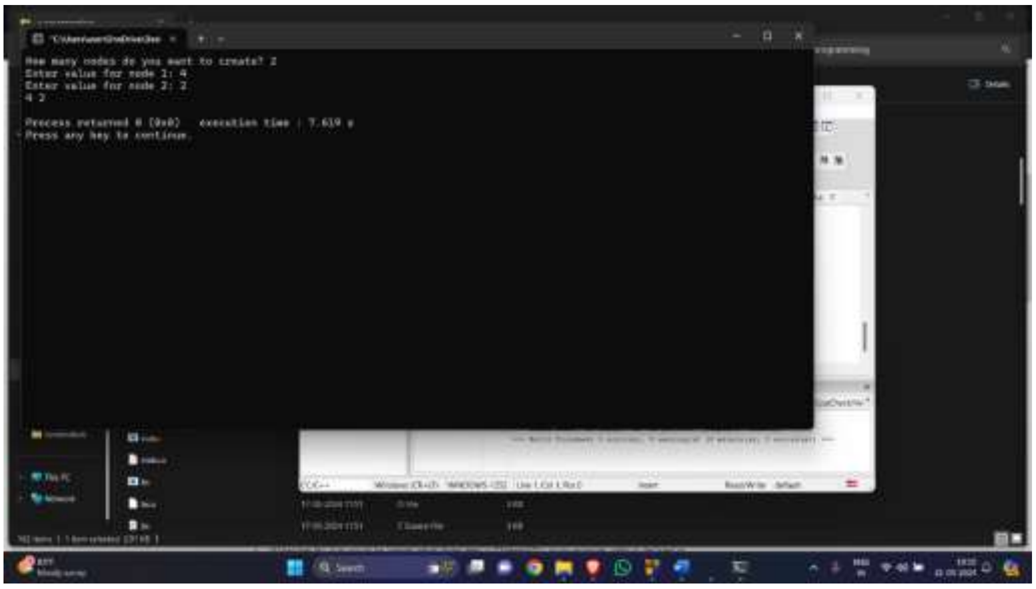
5. Linked list node adding

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

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

void printLinkedlist(struct node *p)
{
  while (p != NULL)
    {
    printf("%d ", p->value);
    p = p->next;
    }
  printf("\n");
}
```

```
int main()
{
 int n, i, value;
 struct node *head = NULL;
 struct node *temp = NULL;
 struct node *p = NULL;

 printf("How many nodes do you want to create? ");
 scanf("%d", &n);

 for (i = 0; i < n; i++)
   {
   temp = (struct node *)malloc(sizeof(struct node));
   if (temp == NULL)
   {
     printf("Memory allocation failed\n");
     return 1;
   }
   printf("Enter value for node %d: ", i + 1);
   scanf("%d", &value);

   temp->value = value;
   temp->next = NULL;

   if (head == NULL)
       {
     head = temp;
       } else {
     p->next = temp;
   }
   p = temp;
 }
```

```
  printLinkedlist(head);


  temp = head;
  while (temp != NULL)
    {
    p = temp->next;
    free(temp);
    temp = p;
    }


  return 0;
}
```

OUTPUT:



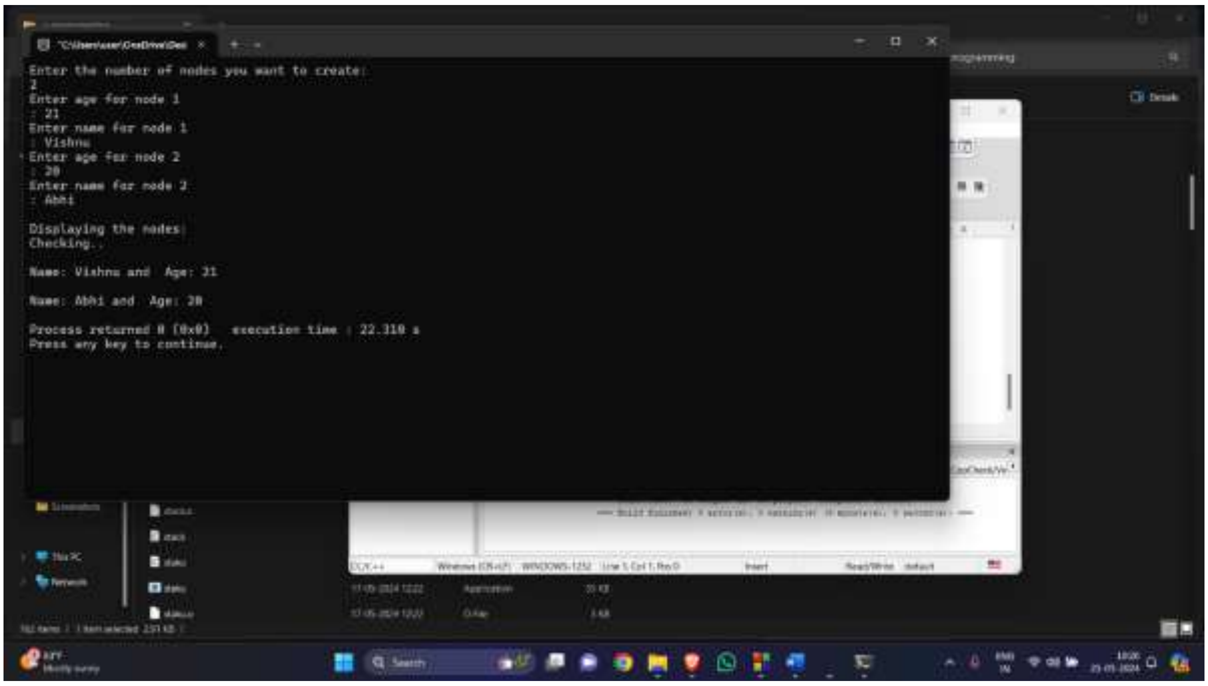
```
How many nodes do you want to create? 2
Enter value for node 1: 4
Enter value for node 2: 2
4 2

Process returned 0 (0x0)   execution time : 7.619 s
Press any key to continue.
```

6. Linked list node creation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Node {
   char name[20];
   int age;
   struct Node *next;
};
```

```
struct Node *head;
void insert(char name[], int age) {
    struct Node *h = head;
    struct Node *nn = (struct Node*)malloc(sizeof(struct Node));
    nn->age = age;
    strcpy(nn->name, name);
    nn->next = NULL;
    if (h == NULL) {
        head = nn;
        return;
    }
    while (h->next != NULL) {
        h = h->next;
    }
    h->next = nn;
}
void display() {
    printf("Checking..\n");
    struct Node* h = head;
    while (h != NULL) {
        printf("\nName: %s and  Age: %d\n", h->name, h->age);
        h = h->next;
    }
}
int main() {
    printf("Enter the number of nodes you want to create:\n");
    int n;
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        int age;
        char name[20];
        printf("Enter age for node %d\n: ", i);
        scanf("%d", &age);
        printf("Enter name for node %d\n: ", i);
        scanf("%s", name);
        insert(name, age);
    }
    printf("\nDisplaying the nodes:\n");
    display();
    return 0;
}
```

OUTPUT:



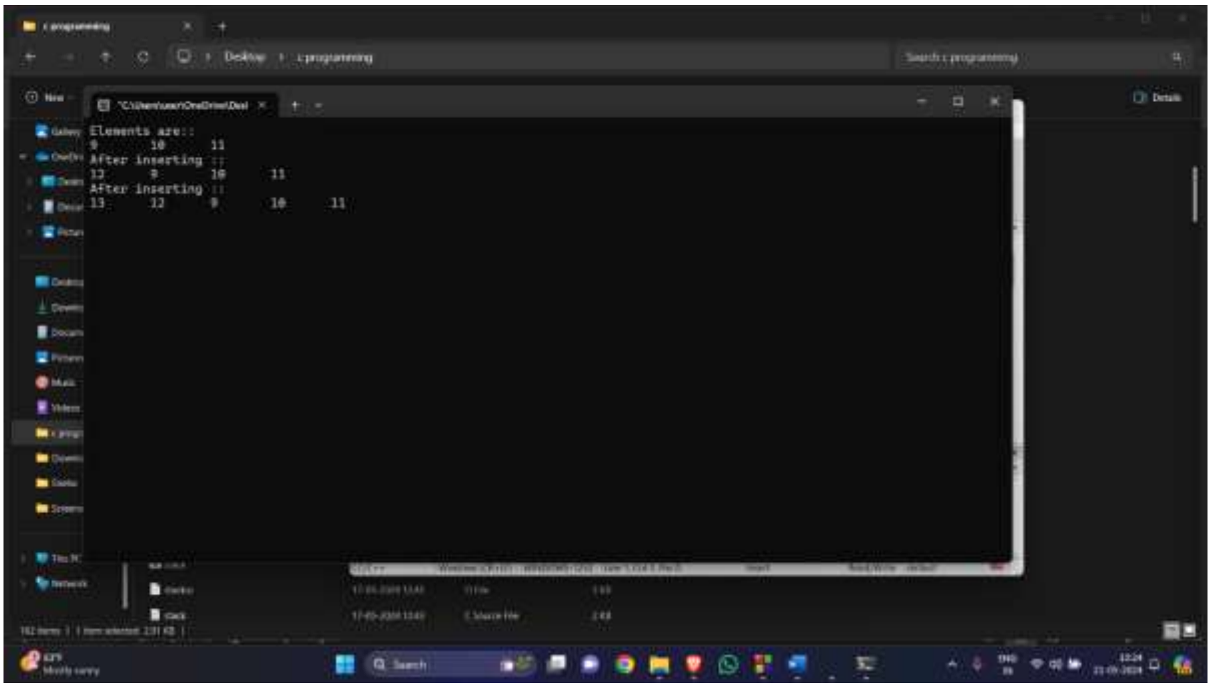
```
Enter the number of nodes you want to create:
2
Enter age for node 1
: 21
Enter name for node 1
: Vishnu
Enter age for node 2
: 20
Enter name for node 2
: Abhi

Displaying the nodes:
Checking..

Name: Vishnu and  Age: 21

Name: Abhi and  Age: 20

Process returned 0 (0x0)   execution time : 22.310 s
Press any key to continue.
```

7. Linked list data insertion

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
struct node{
    int data;
    struct node *link;
    };
```

```
void insert(struct node **head, int data)
{
    struct node *newnode = (struct node *) malloc (sizeof (struct node));
    newnode->data = data;
    newnode->link = *head;
    *head = newnode ;
}
void display (struct node *Node)
{
    while (Node != NULL)
    {
        printf ("%d\t", Node->data);
        Node = Node->link;
    }
    printf("\n");
}
main()
{
    struct node *head = NULL;
    struct node *node2 = NULL;
    struct node *node3 = NULL;
    head = (struct node *) malloc (sizeof (struct node));
    node2 = (struct node *) malloc (sizeof (struct node));
    node3 = (struct node *) malloc (sizeof (struct node));
    head->data = 9;
    head->link = node2;
    node2->data = 10;
    node2->link = node3;
    node3->data = 11;
    node3->link = NULL;
    printf("Elements are:: \n");
    display (head);
    insert(&head, 12);
    printf ("After inserting ::\n");
    display (head);
    insert(&head, 13);
    printf ("After inserting ::\n");
    display (head);
getch();
}
```

OUTPUT:



```
Elements are::
9       10      11
After inserting ::
12      9       10      11
After inserting ::
13      12      9       10      11
```