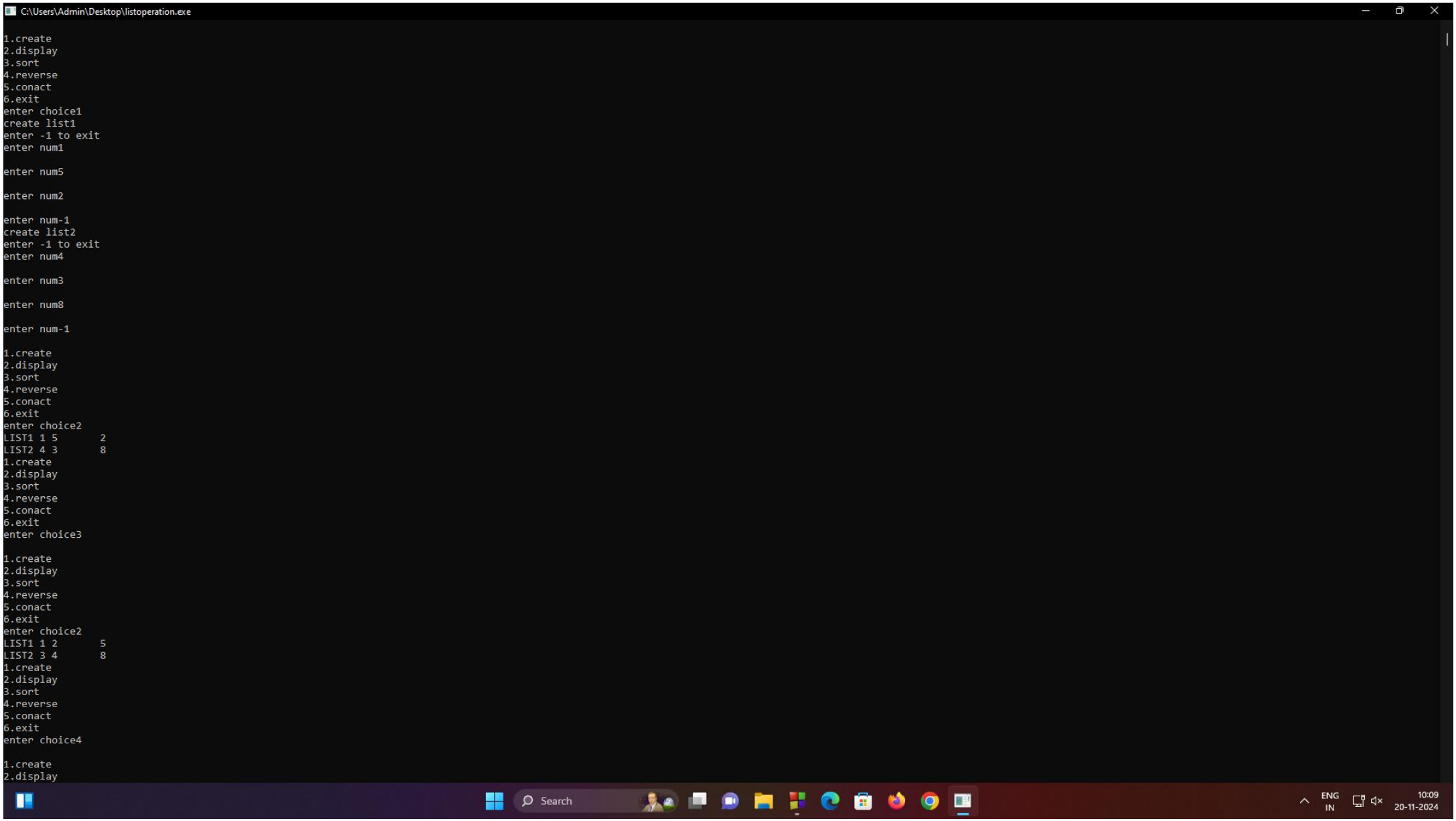
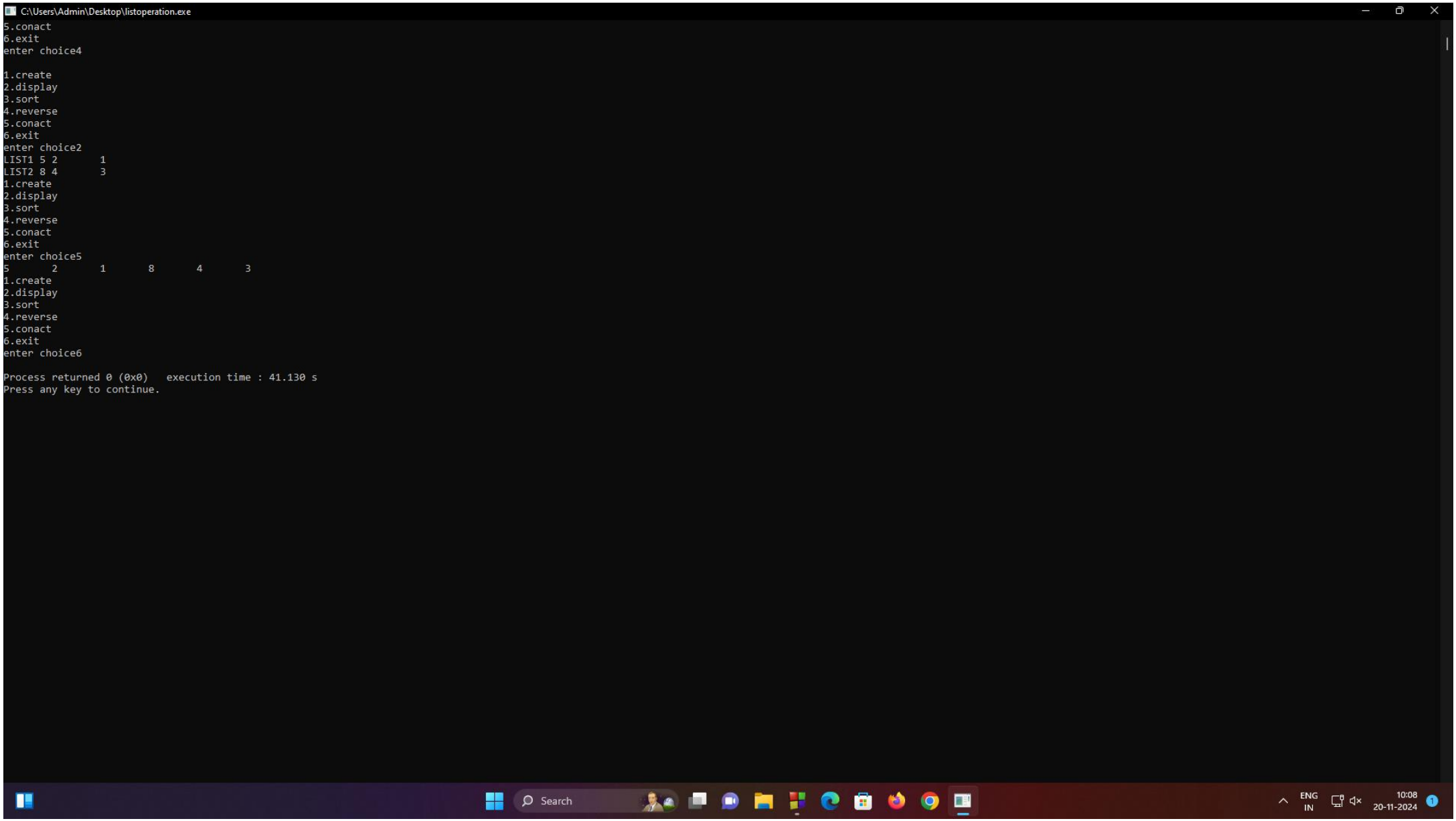
```c
#include <stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node * next;
};
 struct node * start1=NULL;
 struct node * start2=NULL;
 struct node * create(struct node * );
 struct node * display(struct node *);
 struct node * sort(struct node *);
 struct node * rev(struct node *);
 struct node * concat(struct node * start1,struct node * start2);
 void main()
 {
     int choice;
     while(1)
     {   printf("\n1.create\n2.display\n3.sort\n4.reverse\n5.conact\n6.exit");
         printf("\nenter choice");
         scanf("%d",&choice);
        switch(choice)
        {
        case 1:printf("create list1 ");
            start1=create(start1);
            printf("create list2 ");
            start2=create(start2);
            break;
        case 2:printf("LIST1 ");
            start1=display(start1);
            printf("\nLIST2 ");
            start2=display(start2);
            break;
        case 3:
            start1=sort(start1);
            start2=sort(start2);
            break;
        case 4:
            start1=rev(start1);
            start2=rev(start2);
            break;
        case 5:
            start1=concat(start1,start2);
            break;
        case 6:
            exit(0);
        }
```

```
49
50          }
51      }
52      struct node * create(struct node * start)
53      {
54          int num;
55          struct node * newnode ,*ptr;
56
57              printf("\nenter -1 to exit");
58              printf("\nenter num");
59              scanf("%d",& num);
60              while(num!=-1)
61              {
62                  newnode=(struct node * )malloc (sizeof(struct node *));
63                  newnode->data=num;
64                  newnode->next=NULL;
65                  if(start==NULL)
66                     start=newnode;
67                  else
68                  {
69                      ptr=start;
70                      while(ptr->next!=NULL)
71                         ptr=ptr->next;
72                      ptr->next=newnode;
73                  }
74                  printf("\nenter num");
75                  scanf("%d",&num);
76
77              }
78              return start;
79
80      };
81      struct node * display(struct node * start)
82      {
83          struct node * ptr;
84          ptr=start;
85          while(ptr!=NULL)
86          {
87              printf("%d\t",ptr->data);
88              ptr=ptr->next;
89          }
90          return start;
91      };
92       struct node * sort(struct node * start)
93       {
94           struct node * ptr1,*ptr2;
95           ptr1=start;
```

```
97      while(ptr1->next!=NULL)
98      {
99          ptr2=ptr1->next;
100         while(ptr2!=NULL)
101         {
102             if(ptr1->data>ptr2->data)
103             {
104                 temp=ptr1->data;
105                 ptr1->data=ptr2->data;
106                 ptr2->data=temp;
107             }
108             ptr2=ptr2->next;
109         }
110          ptr1=ptr1->next;
111     }
112     return start;
113 };
114
115  struct node * rev(struct node * start)
116  {
117      struct node * ptr=start;
118      struct node * prev=NULL;
119      struct node * next=NULL;
120      while(ptr!=NULL)
121      {
122          next=ptr->next;
123          ptr->next=prev;
124          prev=ptr;
125          ptr=next;
126
127      }
128      start=prev;
129      return prev;
130  };
131
132  struct node * concat(struct node * start1,struct node * start2)
133  {
134      struct node * ptr1=start1;
135       struct node * ptr2=start2;
136       if (ptr1==NULL || ptr2==NULL)
137
138       {
139           printf("one of them is empty");
140           return;
141       }
142       else{
143           while(ptr1->next!=NULL)
```

```
112         return start;
113     };
114
115     struct node * rev(struct node * start)
116     {
117         struct node * ptr=start;
118         struct node * prev=NULL;
119         struct node * next=NULL;
120         while(ptr!=NULL)
121         {
122             next=ptr->next;
123             ptr->next=prev;
124             prev=ptr;
125             ptr=next;
126
127         }
128         start=prev;
129         return prev;
130     };
131
132     struct node * concat(struct node * start1,struct node * start2)
133     {
134         struct node * ptr1=start1;
135          struct node * ptr2=start2;
136          if (ptr1==NULL || ptr2==NULL)
137
138          {
139              printf("one of them is empty");
140              return;
141          }
142          else{
143              while(ptr1->next!=NULL)
144                  ptr1=ptr1->next;
145              ptr1->next=ptr2;
146              display(start1);
147          }
148     };
149
```


C:\Users\Admin\Desktop\listoperation.exe
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice1
create list1
enter -1 to exit
enter num1
enter num5
enter num2
enter num-1
create list2
enter -1 to exit
enter num4
enter num3
enter num8
enter num-1
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice2
LIST1 1 5 2
LIST2 4 3 8
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice3
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice2
LIST1 1 2 5
LIST2 3 4 8
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice4
1.create
2.display
Search
ENG
IN
10:09
20-11-2024


C:\Users\Admin\Desktop\listoperation.exe
5.conact
6.exit
enter choice4
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice2
LIST1 5 2 1
LIST2 8 4 3
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice5
5 2 1 8 4 3
1.create
2.display
3.sort
4.reverse
5.conact
6.exit
enter choice6
Process returned 0 (0x0) execution time : 41.130 s
Press any key to continue.
Search
ENG
IN
10:08
20-11-2024