Source Code:

```
#include <iostream>
using namespace std;
struct newnode{
  int data;
  newnode* next;
};
****************************************
void InsertAtBeg(newnode*& start, int value){
  newnode* newNode= new newnode;
  if(!newNode){
    cout<<"Memory is not allocated"<<endl;
    exit(0);
  }
  newNode->data= value;
  newNode->next= start;
  start= newNode;
}
****************************************
void InsertAtEnd(newnode*& start, int value){
  newnode* newNode= new newnode;
  if(!newNode){
    cout<<"Memory not allocated"<<endl;
    exit(0);
}
  newNode->data= value;
  newNode->next= NULL;
  if(start== NULL){
    start= newNode;
    return;
}
  newnode* ptr= start;
  while(ptr->next != NULL){
    ptr= ptr->next;
  }
  ptr->next= newNode;
}
****************************************
void InsertAfter(newnode*& start, int myData,
int value){
  newnode* newNode= new newnode;
  if(!newNode){
    cout<<"Memory not allocated"<<endl;
    exit(0);
  }
  newNode->data= value;
  newNode->next= NULL;

  newnode* ptr= start;
  while(ptr != NULL && ptr->data != myData){
    ptr= ptr->next;
  }
  if(ptr== NULL){
    cout<<"your searched node not
found."<<endl;
    delete newNode;
    return;
  }
  newNode->next= ptr->next;
  ptr->next= newNode;
}
****************************************
void InsertBefore(newnode*& start, int myData,
int value){
  newnode* newNode= new newnode;
  if(!newNode){
    cout<<"Memory not allocated"<<endl;
```

```
    exit(0);
  }
  newNode->data= value;
  newNode->next= NULL;
  if(start== NULL){
    cout<<"List is empty."<<endl;
    delete newNode;
    return;
  }
  if(start->data== myData){
    newNode->next= start;
    start= newNode;
    return;
  }
  newnode* ptr= start;
  newnode* preptr= ptr;
  while(ptr != NULL && ptr->data != myData){
    preptr= ptr;
    ptr= ptr->next;
  }
  if(ptr== NULL){
    cout<<"your searched node not
found."<<endl;
    delete newNode;
    return;
  }
  preptr->next= newNode;
  newNode->next= ptr;
}
*****************************************
void InsertAtNthPos(newnode*& start, int
value, int position){
  newnode* newNode= new newnode;
  if(!newNode){
    cout<<"Memory not allocated"<<endl;
    exit(0);
  }
  newNode->data= value;
  newNode->next= NULL;
  if(position== 1){
    newNode->next= start;
    start= newNode;
    return;
  }
  newnode* ptr= start;
  for(int i= 1; i < position - 1 && ptr != NULL;
i++){
    ptr= ptr->next;
  }
  if(ptr== NULL){
    cout<<"Position out of bounds."<<endl;
    delete newNode;
    return;
  }
  newNode->next= ptr->next;
  ptr->next= newNode;
}
*****************************************
void deletefrombeg(newnode*& start){
  if(start== NULL){
    cout<< "List is empty."<<endl;
    return;
  }
  newnode* temp= start;
  start= start->next;
  delete temp;
}
```

```
*****************************************
void deletefromend(newnode*& start){
  if(start== NULL){
    cout<< "List is empty."<<endl;
    return;
  }
  newnode* ptr= start;
  newnode* preptr = start;
  while(ptr->next!= NULL){
    preptr= ptr;
    ptr= ptr->next;
  }
  preptr->next= NULL;
  delete ptr;
}
*****************************************
void deleteafter(newnode*& start, int myData){
  if(start== NULL){
    cout<< "List is empty."<<endl;
    return;
  }
  newnode* ptr= start;
  while(ptr != NULL && ptr->data != myData){
    ptr= ptr->next;
  }
  if(ptr== NULL || ptr->next == NULL){
    cout<<"Node not found."<<endl;
    return;
  }
  newnode* temp = ptr->next;
  ptr->next = temp->next;
  delete temp;
}
*****************************************
void deleteNthpos(newnode*& start, int
position){
  if(start== NULL){
    cout<< "List is empty."<<endl;
    return;
  }
  if(position == 1){
    newnode* temp = start;
    start = start->next;
    delete temp;
    return;
  }
  newnode* ptr = start;
  for(int i = 1; i < position - 1 && ptr != NULL;
i++){
    ptr = ptr->next;
  }
  if(ptr == NULL || ptr->next == NULL){
    cout<<"Position out of bounds."<<endl;
    return;
  }
  newnode* temp = ptr->next;
  ptr->next = temp->next;
  delete temp;
}
*****************************************
void display(newnode* start){
  if(start == NULL){
    cout << "List is empty." << endl;
    return;
  }
  newnode* temp = start;
  while(temp != NULL){
```

```
    cout << temp->data << " ";
    temp = temp->next;
  }
  cout << endl;
}
*****************************************
int main(){
  newnode* start= NULL;
  cout<<"choose a option:"<<endl;
  cout<<"1. Insert at beginning"<<endl;
  cout<<"2. Insert at end"<<endl;
  cout<<"3. Insert after a node"<<endl;
  cout<<"4. Insert before a node"<<endl;
  cout<<"5. Insert at nth position"<<endl;
  cout<<"6. delete from beginning"<<endl;
  cout<<"7. delete from end"<<endl;
  cout<<"8. delete after a node"<<endl;
  cout<<"9. delete at nth position"<<endl;
  cout<<"10. Display the list"<<endl;
  cout<<"11. Exit"<<endl;
  int choice, value, myData, position;
  while(true){
    cout<<"Enter your choice: ";
    cin>>choice;
    switch(choice){
      case 1:
        cout<<"Enter value to insert at
beginning: ";
        cin>>value;
        InsertAtBeg(start, value);
        break;
      case 2:
        cout<<"Enter value to insert at end: ";
        cin>>value;
        InsertAtEnd(start, value);
        break;
      case 3:
        cout<<"Enter value to insert after which
node: ";
        cin>>myData;
        cout<<"Enter value to insert: ";
        cin>>value;
        InsertAfter(start, myData, value);
        break;
      case 4:
        cout<<"Enter value to insert before
which node: ";
        cin>>myData;
        cout<<"Enter value to insert: ";
        cin>>value;
        InsertBefore(start, myData, value);
        break;
      case 5:
        cout<<"Enter position to insert at: ";
        cin>>position;
        cout<<"Enter value to insert: ";
        cin>>value;
        InsertAtNthPos(start, value, position);
        break;
      case 6:
        deletefrombeg(start);
        break;
      case 7:
        deletefromend(start);
        break;
      case 8:
        cout<<"Enter value after which node to
delete: ";
```

```
            cin>>myData;
            deleteafter(start, myData);
            break;
        case 9:
            cout<<"Enter position to delete at: ";
            cin>>position;
            deleteNthpos(start, position);
            break;
        case 10:
            display(start);
            break;
        case 11:
            cout<<"Exiting..."<<endl;
            break;
        default:
            cout<<"Invalid choice. Please try
again."<<endl;
      }
      if(choice == 11) {
        break;
      }
   }
   return 0;
}
```

Output:

```
choose a option:
1. Insert at beginning
2. Insert at end
3. Insert after a node
4. Insert before a node
5. Insert at nth position
6. delete from beginning
7. delete from end
8. delete after a node
9. delete at nth position
10. Display the list
11. Exit
```

```
Enter your choice: 1
Enter value to insert at beginning: 30
Enter your choice: 10
30
Enter your choice: 1
Enter value to insert at beginning: 10
Enter your choice: 10
10 30
Enter your choice: 2
Enter value to insert at end: 60
Enter your choice: 10
10 30 60
Enter your choice: 3
Enter value to insert after which node: 30
Enter value to insert: 23
Enter your choice: 10
10 30 23 60
Enter your choice: 4
Enter value to insert before which node: 60
Enter value to insert: 76
Enter your choice: 10
10 30 23 76 60
Enter your choice: 5
Enter position to insert at: 4
Enter value to insert: 100
Enter your choice: 10
10 30 23 100 76 60
```

```
Enter your choice: 10
10 30 23 100 76 60
Enter your choice: 6
Enter your choice: 10
30 23 100 76 60
Enter your choice: 7
Enter your choice: 10
30 23 100 76
Enter your choice: 8
Enter value after which node to delete: 23
Enter your choice: 10
30 23 76
Enter your choice: 9
Enter position to delete at: 2
Enter your choice: 10
30 76
Enter your choice: 11
Exiting...
```