1

```
#include<iostream>
#include<stdlib.h>
using namespace std;

//Creating node
struct node
{
   int data;
   struct node *next;
};
struct node *head;

//Function declaration
void beginsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();

//Main Function Definition
int main ()
{
   int choice =0;
   while(choice != 9)
   {
      cout<<"\n\n*********Main Menu*********\n";
      cout<<"\nChoose one option from the following list: \n";
      cout<<"\n========================================\n";
      cout<<"\n1.Insert at beginning\n2.Insert at last\n3.Insert at any random location\n";
      cout<<"4.Delete from Beginning\n5.Delete from last\n6.Delete node after specified location\n"
      cout<<"7.Search for an element\n8.Display elements\n9.Exit\n";
      cout<<"\nEnter your choice?\n";
      cin>>choice;
```

2

```
      switch(choice)
      {
         case 1:
         beginsert();
         break;
         case 2:
         lastinsert();
         break;
         case 3:
         randominsert();
         break;
         case 4:
         begin_delete();
         break;
         case 5:
         last_delete();
         break;
         case 6:
         random_delete();
         break;
         case 7:
         search();
         break;
```

3

```
         case 8:
         display();
         break;
         case 9:
         exit(0);
         break;
         default:
         cout<<"Please enter valid choice..";
      }
   }
}
```

11

```
//Displaying the list of nodes
void display()
{
   struct node *ptr;
   ptr = head;
   if(ptr == NULL)
   {
      cout<<"Nothing to print";
   }
   else
   {
      cout<<"\nprinting values . . . . .\n";
      while (ptr!=NULL)
      {
         cout<<ptr->data<<" ";
         ptr = ptr -> next;
      }
   }
}
```

4

```
//Inserting at the beginning of the node
void beginsert()
{
   struct node *ptr;
   int item;
   ptr = (struct node *) malloc(sizeof(struct node *));
   if(ptr == NULL)
   {
      cout<<"\nOVERFLOW";
   }
   else
   {
      cout<<"\nEnter value \n";
      cin>>item;
      ptr->data = item;
      ptr->next = head;
      head = ptr;
      cout<<"\nNode inserted";
   }
}
```

6

```
//Deleting at the beginning of the node
void begin_delete()
{
   struct node *ptr;
   if(head == NULL)
   {
      cout<<"\nList is empty\n";
   }
   else
   {
      ptr = head;
      head = ptr->next;
      delete ptr;
      cout<<"\nNode deleted from the begining ...\n";
   }
}
```

5

```
//Inserting at the end of the node
void lastinsert()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node*)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      cout<<"\nOVERFLOW";
   }
   else
   {
      cout<<"\nEnter value?\n";
      cin>>item;
      ptr->data = item;
      if(head == NULL)
      {
         ptr -> next = NULL;
         head = ptr;
         cout<<"\nNode inserted";
      }
      else
      {
         temp = head;
         while (temp -> next != NULL)
         {
            temp = temp -> next;
         }
         temp->next = ptr;
         ptr->next = NULL;
         cout<<"\nNode inserted";
      }
   }
}
```

7

```
//Inserting after a specified location
void randominsert()
{
   int i,loc,item;
   struct node *ptr, *temp;
   ptr = (struct node *) malloc (sizeof(struct
node));
   if(ptr == NULL)
   {
      cout<<"\nOVERFLOW";
   }
   else
   {
      cout<<"\nEnter element value: ";
      cin>>item;
      ptr->data = item;
      cout<<"\nEnter the location after which you
want to insert: ";
      cin>>loc;
      temp=head;
      for(i=0;i<loc;i++)
      {
         temp = temp->next;
         if(temp == NULL)
         {
            cout<<"\ncan't insert\n";
            return;
         }

      }
      ptr ->next = temp ->next;
      temp ->next = ptr;
      cout<<"\nNode inserted";
   }
}
```

8

```
//Deleting after a specified location
void random_delete()
{
   struct node *ptr,*ptr1;
   int loc,i;
   cout<<"\n Enter the location of the node after
which you want to perform deletion: ";
   cin>>loc;
   ptr=head;
   for(i=0;i<loc;i++)
   {
      ptr1 = ptr;
      ptr = ptr->next;

      if(ptr == NULL)
      {
         cout<<"\nCan't delete";
         return;
      }
   }
   ptr1 ->next = ptr ->next;
   delete ptr;
   cout<<"\nDeleted node at "<<loc+1;
}
```

9

```
//Deleting at the end of the node
void last_delete()
{
    struct node *ptr,*ptr1;
    if(head == NULL)
    {
        cout<<"\nlist is empty";
    }
    else if(head -> next == NULL)
    {
        head = NULL;
         delete head;
        cout<<"\nOnly node of the list deleted ...\n";
    }
    else
    {
        ptr = head;
        while(ptr->next != NULL)
        {
            ptr1 = ptr;
            ptr = ptr ->next;
        }
        ptr1->next = NULL;
        delete ptr;
        cout<<"\nDeleted Node from the last ...\n";
    }
}
```

10

```
//Searching a node
void search()
{
    struct node *ptr;
    int item,i=0,status=-1;
    ptr = head;
    if(ptr == NULL)
    {
        cout<<"\nEmpty List\n";
    }
    else
    {
     cout<<"\nEnter item which you want to search?: ";
        cin>>item;
        while (ptr!=NULL)
        {
            if(ptr->data == item)
            {
                cout<<"item found at location: "<<i;
                status++;
            }

            i++;
            ptr = ptr -> next;
        }

        if(status==-1)
            {

            cout<<"Item not found\n";
             }

    }
}
```