Name: RITHIKA KATHIRVEL
Roll number: B200055CS
Batch: B

Main func
→read the str from user
→ call extract string func

//pseudocode for extract str func

```
        int i;
        int j=0;
        for(i=1;i<(strlen(str)-2);i++)
        {
                if(str[i]!=' ')
                {    //printf("%d %c\n",i,str[i]);
                        if(str[i]=='(' && str[i+1]==' ' && str[i+2]==')')

                                {
                                        str2[j]=str[i];
                                        str2[j+1]='N';
                                        str2[j+2]=str[i+2];
                                        i=i+2;
                                        j=j+3;
                                }
                        else
                                {
                                str2[j]=str[i];
                                j++;
                                }
                }
        }
   str2[j]='\0';
```

```
//pseudocode for tree insert

int index_t(char str[],int start,int end)
{
	struct stack *S;
	int i;
	S=(struct stack*)malloc(sizeof(struct stack));
	S->size=500;
	S->top=-1;
	if(start>end)
		return -1;
	for(i=start;i<=end;i++)
	{
		if(str[i]=='(')
			PUSH(S,str[i]);
		else if(str[i]==')')
			if(S->A[S->top]='(')
				{
				POP(S);
				if(STACK_EMPTY(S)==-1)
					return i;
				}
	}
	return -1;
}

struct node_bt* tree_insert(char str2[], int start, int end)
{


	if(start>end)
		return NULL;
	struct node_bt *root;
	root=(struct node_bt*)malloc(sizeof(struct node_bt));
	int num=0;
	int neg=0;


	int k;
	if(str2[start]=='-')
		{
		neg=1;
		start++;
		//printf("in neg\n");
```

```
                }
        while(str2[start]>='0' && str2[start]<='9')
                {
                num*=10;
                num=num+(str2[start]-'0');
                start++;


                }
        start--;

        if(neg==1)
                num=num*-1;
        //printf("%d ",num);
        if(str2[start+1]=='N')
                root=create_node(-9999999);
        else
                root=create_node(num);
        int i=-1;
        if(start+1<=end && str2[start+1]=='(')
                i=index_t(str2,start+1,end);
        if(i!=-1)
                {
                root->leftchild=tree_insert(str2,start+2,i-1);
                root->rightchild=tree_insert(str2,i+2,end-1);
                }
        return root;


}

//pseudocode for read_and_store()
struct node_bt* root;
root=(struct node_bt*)malloc(sizeof(struct node_bt));
root=NULL;
 root=tree_insert(str_p,0,strlen(str_p)-1);

Return root;

//pseudocode for print_employees(T)
void Preorder(struct node_bt *root) {

        if (root != NULL && root->value!=-9999999) {
```

```
                Preorder(root->leftchild);
                printf("%d ", root->value_id);
                Preorder(root->rightchild);

        }

int isBST(struct node_bt* root){
   struct node_bt *p=NULL;
   if(root!=NULL && root->value!=-9999999){
      if(!isBST(root->leftchild)){
         return 0;
      }
      if(p!=NULL && p->value!=-9999999 && root->value<=p->value){
         return 0;
      }
      p=root;
      return isBST(root->rightchild);
   }
   return 1;

}

//pseudcode to find max_employee(T)
struct node_bst* maximum(struct node_bst* node_test){
  struct node_bst* max;
  max=(struct node_bst*)malloc(sizeof(struct node_bst));
  max=node_test;
  while(max->rightchild!=NULL)
    max=max->rightchild;

 return max;
}


//pseudocode for find_employees(T)
for(int i=0;i<heightof(t);i++)
{if(T->reward>T->leftchil->rewrd+T->right->child->reward)
   print(T)
Else continue;
}
```